Méthode de synthèse d'un contrôleur logique à partir des spécifications algébriques

Antonio Medina Rodriguez

N° ENSC-2007/82

THESE DE DOCTORAT
DE L'ECOLE NORMALE SUPERIEURE DE CACHAN

Présentée par

Monsieur Antonio MEDINA RODRÍGUEZ

pour obtenir le grade de

DOCTEUR DE L'ECOLE NORMALE SUPERIEURE DE CACHAN

Domaine :
ELECTRONIQUE–ELECTROTECHNIQUE–AUTOMATIQUE

Sujet de la thèse :

Méthode de synthèse d'un contrôleur logique à partir des spécifications algébriques

Thèse présentée et soutenue à Cachan le 5 décembre 2007 devant le jury composé de :

Hassane ALLA	Professeur – INPG – GIPSA lab	Rapporteur
Véronique CARRE-MENETRIER	Professeur – URCA – CRESTIC	Rapporteur
Eric RUTTEN	Chargé de Recherche – INRIA-RA	Examinateur
Jean-Marc FAURE	Professeur – SUPRIMECA – LURPA	Directeur de thèse
Jean-Marc ROUSSEL	Maître de conférences–ENS Cachan LURPA	Encadrant

Laboratoire Universitaire de Recherche en Production Automatisée
ENS CACHAN / EA 1385
61, avenue du Président Wilson, 94235 CACHAN CEDEX (France)

Remerciements

Le travail de recherche exposé dans ce mémoire de thèse a été réalisé au sein du Laboratoire Universitaire de Recherche en Production Automatisée (LURPA) de l'École Normale Supérieure de Cachan.

Les conseils et soutiens tantôt éducatif et tantôt personnels de tant de personnes, collègues, camarades de classe et amis, leur amitié ou tout simplement leur présence, m'ont permis de concrétiser cette thèse. Pour chacun de ceux qui m'ont aidé d'une manière ou d'une autre sur ce chemin de ma vie, je tiens à exprimer mes plus profonds et sincères remerciements.

Tout d'abord, je tiens à remercier mon directeur de thèse le Professeur Jean-Marc FAURE qui a assuré la direction de ce travail de recherche. Les mots ne suffisent pas pour exprimer mes plus profonds et sincères remerciements, ainsi que la reconnaissance et l'estime que j'ai pour ses conseils scientifiques avisés et pour sa grande disponibilité personnelle ainsi que par sa qualité humaine, le soutien et la confiance qu'il a exprimée à mon égard et pour mon travail au cours des ces cinq dernières années ; mais surtout parce qu'il m'a aidé lorsque j'en avait le plus besoin.

Je tiens à remercier et à exprimer ma gratitude à Jean-Marc ROUSSEL pour avoir accepté d'encadré mes recherches. Malgré les difficultés qui ont pu surgir durant ces années, il a toujours su montrer sa grande volonté de me soutenir à travers ses points de vue scientifiques avisés.

Je tiens également à exprimer mes très vifs remerciements à Monsieur Hassane ALLA, pour m'avoir fait l'honneur d'être membre du Jury comme rapporteur de ma thèse. Je lui exprime ma plus vive reconnaissance pour ses qualités humaines et ses conseils. Je tiens ici à le remercier à nouveau pour avoir accepté de diriger mon mémoire de DEA au LAG du Grenoble.

Enfin, je tiens à exprimer également ma profonde gratitude et remerciements envers Madame Véronique CARRÉ- MENETRIER et Monsieur Eric RUTTEN pour m'avoir accordé le privilège d'accepter d'être respectivement rapporteur et examinateur de ce travail de recherche.

La réalisation de cette thèse a été rendue possible grâce au soutien financier du **Consejo Nacional de Ciencia y Tecnología (CONACYT)** -Conseil *National de la Science et technologie du Mexique*- et aussi par le suivi académique et le soutien de la Société Française d'Exportation des Ressources Éducatives (SFERE), et notamment Madame Anna MANETA, qui a soutenu mon transfert au LURPA.

Je tiens à remercier tous les membres du laboratoire, permanents et étudiants, que j'ai eu le privilège de rencontrer durant mon séjour au LURPA. Je témoigne ma plus profonde gratitude à Marc JACHYM, à son épouse, à sa famille, mais aussi à Charyar et Sylvain pour leur

grande qualité humaine et pour leur soutien inconditionnel qui a été une des clés pour finir ma thèse. Pour toute votre attention, pour votre amitié, pour tout cela et plus encore, de tout mon cœur : un grand merci.

A mes amis et compatriotes, Zulema et Israel, qui m'ont accompagné dans cette aventure dans les bons et les mauvais moments, au laboratoire ou à la fête, sincèrement et vivement : merci pour votre soutien. Merci aussi pour vos encouragements, pour votre confiance et parce que je sais que vous vous êtes réjouis autant que moi de la fin de cette étape. Je remercie également mes collègues et amis : Steve, Gaëlle, Yann, Vincent, Silvain, Mathieu et tous les autres étudiants et enseignants, qui par leurs qualités humaines ont toujours fait du laboratoire un lieu agréable où travailler, muchas gracias !

À tous ceux ayant contribué de près ou de loin à l'élaboration de ce travail et que j'ai oublié de mentionner mais qui se reconnaîtront, je les remercie.

Dans un contexte plus personnel, je remercie tous les gens que j'ai rencontré en France et qui m'ont offert leur amitié : toute la bande de Grenoble sans oublier les "chipocles románticos", les amis de la Maison du Mexique, du Ballet Folklorique et tous les autres. Aschaf, Rosemberg, Catalina, Hector, Luis, Paloma, Javier Campo, Christine Azevedo, Soraya, Fabienne, Anne, Irina, Isis, Alvaro, Eloy, Aleyda, Marcela, Norberto, Antonio, Maru, Enrique, Edgar, Jorge, Jorgito del área 51, Théa, Marieke et tous ceux que je ne peux malheureusement pas citer dans cet espace mais qui se reconnaîtront facilement ici. À vous tous je vous exprime ma plus totale gratitude. Merci enfin a Nidiyare pour m'avoir donné le plus beau des cadeaux.

A mis amigos que aun estando en México siempre me han acompañado con el corazón: Nicolás, Isidro, Sarita, Chelo, Olga y la familia Lugo, les agradezco infinitamente su sincera amistad.

A mis papas, Ernestino y Candelaria, a mis hermanos Paco, Consuelo, Neto y Bety, que siempre han sido todo un ejemplo para mí y que han estado en mi pensamiento y corazón, por todo lo que representan para mí, les agradezco desde el fondo de mi alma. Esta tesis se las dedico porque representa el logro, el esfuerzo y el cariño de todos ustedes. Especialmente a Neto, que nunca podré agradecerle suficientemente todo el apoyo que me brindó sin restricciones.

Antonio MEDINA RODRIGUEZ

A mis papás,

Ernestino y Candelaria con todo mi amor y respeto.

À Nicté-Ha,

Sans le vouloir en déménageant au Mexique tu es partie avec un morceau de mon cœur mais dans l'autre morceau j'ai gardé tous les bons moments pour y tenir.

NO TE RINDAS

Cuando sientas que todo es cuesta arriba,
cuando creas que no hay salida,
cuando el cansancio sea inmenso,
descansa si quieres pero... NO TE
RINDAS.

Cuando la nube sea tan gris,
que parezca que no hay luz,
y parece que no hay salida,
llora si quieres pero...
NO TE RINDAS.

Porque después de la desesperación,
después de toda tormenta,
después de descargar tu alma,
veras que valió la pena...
pero... NO TE RINDAS.

Cuando tus fuerzas se agoten,
cuando el desánimo te arrastre,
la confianza se termina,
alienta, hay esperanza…..NO TE
RINDAS.

Cuando los días vayan pasando,
uno a uno, tú contándolos,
verás que la lucha compensa,
sigue pero……..NO TE RINDAS.

MARIBEL FREY

NE RENONCE PAS

Quand tu sens que tout est en montée,
quand tu penses qu'il y a aucune sortie,
quand la fatigue est immense,
repose-toi si tu veux, mais ... NE
RENONCE PAS.

Quand le nuage est si gris,
qu'il semble qu'il n'y a pas de lumière,
et il semble qu'il n'y a pas de sortie,
pleure si tu veux, mais ...
NE RENONCE PAS.

Parce qu'après le désespoir,
après tout tempête,
après avoir déchargé ton âme,
tu verras que cela a valu la peine ...
mais ... NE RENONCE PAS.

Quand tes forces sont épuisées,
quand le découragement s'installe,
la confiance prend fin,
courage, il y a de l'espoir ..... NE
RENONCE PAS.

Quand les jours passent,
un par un, toi en train de les compter,
tu verras que la lutte compense,
continue mais ..... NE RENONCE PAS.

MARIBEL FREY

Table de matières

Table des équations

$$\left\{\begin{array}{l}\left\{\begin{array}{ll}1a) & s \le X\\ 2a) & r \le \overline{X}\\ 3a) & m \le REP_X\end{array}\right.\\ \left\{\begin{array}{ll}1a) & k_s \le X\\ 2a) & k_r \le \overline{X}\\ 3a) & k_m \le REP_X\end{array}\right.\\ \left\{\begin{array}{ll}hyp_1) & s\cdot r + s\cdot m + r\cdot m = 0 *\\ hyp_2) & k_s\cdot k_r + k_s\cdot k_m + k_r\cdot k_m = 0 *\\ hyp_3) & \left(\begin{array}{r} s\cdot(k_r+k_m)+r\cdot(k_s+k_m)\\ +m\cdot(k_s+k_r)\end{array}\right) = 0 *\\ hyp_4) & \bar{s}\cdot\bar{r}\cdot\bar{m}\cdot\overline{k_s}\cdot\overline{k_r}\cdot\overline{k_m} = 0 *\end{array}\right.\end{array}\right. \Leftrightarrow \begin{array}{c} X = SR(s+k_s,\, r+k_r)\\ \text{ou bien}\\ X = RS(s+k_s,\, r+k_r)\end{array} \quad .....[5.1] \quad ..... 86$$

$$\left\{\begin{array}{ll}1a) & s \le X\\ 2a) & r \le \overline{X}\\ 3a) & \bar{s}\cdot\bar{r} \le REP_X\\ cohérence) & s\cdot r = 0 *\end{array}\right. \Leftrightarrow \begin{array}{c} X = SR(s,r)\\ \text{ou bien}\\ X = RS(s,r)\end{array} \quad .....[5.2] \quad ..... 86$$

$$\left\{\begin{array}{ll}1a) & s \le X\\ 2a) & \bar{s}\cdot r \le \overline{X}\\ 3a) & \bar{s}\cdot\bar{r} \le REP_X\end{array}\right. \Leftrightarrow X = SR(s,r) \quad .....[5.3] \quad ..... 86$$

$$\left\{\begin{array}{ll}1a) & s\cdot\bar{r} \le X\\ 2a) & r \le \overline{X}\\ 3a) & \bar{s}\cdot\bar{r} \le REP_X\end{array}\right. \Leftrightarrow X = RS(s,r) \quad .....[5.4] \quad ..... 87$$

Table des figures

Introduction

Les tâches liées au processus de conception des systèmes automatisées (spécification, conception, implantation) qui étaient auparavant basées sur l'expérience et le savoir-faire, sont devenues de plus en plus difficiles à maîtriser dans les systèmes industriels d'aujourd'hui à cause de leur taille et complexité. En effet, les avancées technologiques comme de nouveaux capteurs, actionneurs, l'apparition de l'Automate Programmable Industriel (API), d'ordinateurs et la croissance de la puissance et vitesse de calcul de ceux-ci permettent de contrôler des systèmes très larges et complexes.

Pourtant, ces systèmes sont devenus de plus en plus critiques pour la sécurité des biens et des personnes à cause de la nature du processus qu'ils contrôlent (processus chimiques, centrales thermiques et nucléaires...). Ceci, lié à la demande croissante de la société pour la prévention des risques industriels, explique que de plus en plus d'entreprises souhaitent disposer de méthodes de développement rigoureuses permettant de garantir que le comportement de ces systèmes est bien celui requis, et ne contient pas d'éléments nuisibles à la sûreté. C'est la raison pour laquelle nous plaçons nos travaux dans le cadre de la sûreté de fonctionnement.

Les techniques de synthèse formelle visent à garantir à priori la sûreté de fonctionnement en produisant automatiquement (ou semi-automatiquement) un modèle de commande respectant les propriétés de vivacité (ce qu'il faut faire), de sécurité (ce qu'il ne faut pas faire), et éventuellement de célérité (contraintes temporelles) requises par l'application.

De nombreuses recherches ont été réalisées dans ce domaine ces vingt dernières années à partir notamment des travaux de Wonham [Won 87] à l'Université de Toronto. Quel que soit l'intérêt de ce cadre théorique, dit de la commande supervisée, il n'en demeure pas moins vrai que son application à des systèmes industriels reste encore à démontrer.

Ceci provient de notre point de vue principalement de problèmes d'explosion combinatoire qui sont inhérents à cette méthode de synthèse et qui se produisent lorsqu'on l'applique à des systèmes industriels, même de taille réduite. En fait, les problèmes d'explosion combinatoire se rencontrent chaque fois que l'on manipule des automates représentatifs d'un système réel.

Un autre inconvénient de ces travaux est qu'ils se contentent le plus souvent de générer un modèle conceptuel de commande, sous forme d'automate à états, et ne s'intéressent pas toujours à l'implantation de ce modèle sur API. Or, une démarche de synthèse formelle doit, pour être applicable dans l'industrie, couvrir la phase de développement des programmes, à l'aide de langages métier tels que ceux de la norme IEC 61131-3 (Instruction List, Structured Text, Ladder Diagram, Function Block Diagram, Sequential Function Chart).

Ceci a induit des chercheurs du LURPA à explorer une autre voie pour la vérification de contrôleurs : la modélisation des comportements sous forme algébrique. Dans cette approche, le programme et les propriétés à vérifier sont tous les deux exprimés sous forme d'un jeu d'équations dans une algèbre convenable. La preuve de propriétés est alors effectuée par un raisonnement symbolique dont l'objectif est de démontrer le jeu d'équations représentant les propriétés à partir de celui caractérisant le programme. Afin de tenir compte des spécificités des systèmes de commande industriels qui conduisent à la manipulation conjointe d'états de variables logiques, d'événements correspondant à des changements d'états de ces variables, et d'expressions fonctions du temps physique, une algèbre originale, dénommée $\mathbb{I}$, a été proposée car elle fournit un cadre pour l'intégration de ces trois entités [Rou_03a].
Les résultats des travaux relatifs à l'analyse formelle de programmes API nous ont conduit à explorer une approche basée sur cette algèbre afin de chercher une alternative intéressante aux méthodes conventionnelles de synthèse formelle basées sur la théorie de la commande supervisée et utilisant une représentation comportementale par automate. Plus précisément, nous avons souhaité développer une méthode de synthèse formelle utilisant en lieu et place d'une modélisation par automates à états une représentation algébrique des propriétés à satisfaire, l'algèbre utilisée étant l'algèbre $\mathbb{I}$.

D'autre part, la théorie de la commande supervisée ne considère pas la façon d'obtenir la spécification formelle. Nous considérons cependant que ce problème d'expression des spécifications ne peut pas être négligé. Nous chercherons donc à prendre en compte les besoins et objectifs de l'utilisateur au travers de la spécification de fonctionnement.

L'objectif de travail de thèse présenté dans ce document est donc de développer une méthode de synthèse de la commande, alternative à la technique de synthèse de la commande supervisée, en partant de la spécification de la commande en langage naturel (LN) **jusqu'à l'implantation des fonctions de commande générées par la méthode dans un API**.

Nous utiliserons, pour développer cette méthode un cadre mathématique formel qui est utilisé dès la formalisation de la spécification jusqu'à l'implantation, afin d'assurer la cohérence de formalisme dans la méthode de synthèse proposée.

Nous cherchons à détecter le plutôt possible les incohérences menant à une absence de solution, ceci afin d'éliminer ces incohérences en faisant appel à l'utilisateur. Notre objectif n'est pas seulement de savoir qu'il y a une erreur dans la spécification mais aussi de l'identifier afin qu'elle soit corrigée.

Le présent mémoire de thèse est organisé comme suit :

Le premier chapitre est consacré à la présentation du contexte de nos travaux : la synthèse de la commande des Systèmes à Événements Discrets (SED). La sûreté de la commande étant un de nos soucis, nous ferons le positionnement de nos travaux par rapport à celle-ci. Nous décrirons le système de contrôle-commande étudié ainsi que la théorie de synthèse de la commande introduite par Ramadge et Wonham dans les années 80. Nous présentons ensuite une analyse des limitations actuelles pour l'application de cette théorie. A la fin du chapitre nous présentons l'objectif de nos propositions.

Les deux chapitres suivants présentent le cadre formel algébrique sur lequel s'appuie notre méthode de synthèse. Les bases de l'algèbre $\mathbb{I}$ sont présentées au chapitre 2 : les opérations ET, OU et NON, la relation d'égalité et la relation d'ordre partiel "≤" (appelé aussi inclusion). Cette dernière relation est un point clef dans la méthode proposée. Le chapitre 3 est dédié aux opérations temporelles définies dans la norme [IEC 61131-3] : les opérations Fronts, les opérations Mémoires et les opérations Temporisateurs.

Dans le chapitre 4, nous introduisons une classification des spécifications qui servira par la suite. Ce chapitre est aussi dédié à l'obtention de spécifications formelles à partir d'énoncés en langage naturel (LN) en formalisant trois exemples.

Dans le chapitre 5, nous présentons une technique de résolution d'un système d'équations, technique qui sert de base à la synthèse de fonctions de contrôle.

La méthode de synthèse globale est détaillée dans le chapitre 6. Cette méthode s'appuie sur la vérification algébrique des propriétés du système formel obtenu à partir des spécifications et sur l'intervention du concepteur en cas de non respect de ces propriétés.

Enfin, les chapitres 7 et 8 illustrent l'application de la méthode proposée sur les trois exemples formalisés au chapitre 4.

Chapitre 1

Dans ce chapitre, nous présentons le contexte de nos travaux de recherche sur la synthèse de la commande des Systèmes à Evénements Discrets (SED). Dans un premier temps, nous effectuons une brève description du système de contrôle-commande étudié. Ensuite nous positionnons nos travaux dans le cadre de la sûreté de fonctionnement. Nous présentons ensuite succinctement la théorie de la synthèse de la commande introduite dans les années 80 par Ramadge et Wonham. Puis, nous analysons les limitations actuelles de cette théorie avant de présenter l'objectif de nos propositions.

1. Positionnement des travaux

Les travaux de recherche qui sont rapportés dans ce mémoire concernent la synthèse d'un système de commande logique qui communique avec un système physique au travers de capteurs et d'actionneurs (figure 1). De plus, nous privilégierons une implantation de ce système de commande sur API (automate programmable industriel), équipement qui est très utilisé dans l'industrie.

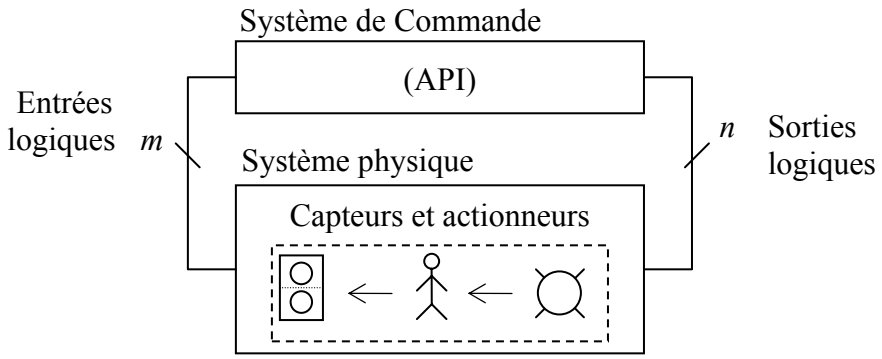


Figure 1 *Système de commande logique*

La conception et l'implantation de ces systèmes de commande, qui étaient auparavant basées sur l'expérience et le savoir-faire, sont devenues de plus en plus difficiles à maîtriser dans les systèmes industriels d'aujourd'hui à cause de leur taille et de leur complexité. Pourtant, ces systèmes de commande sont devenus de plus en plus critiques pour la sécurité de biens et de personnes. Ceci, lié à la demande croissante de la société pour la prévention des risques industriels, explique que de plus en plus d'entreprises souhaitent disposer de méthodes de développement rigoureuses permettant de garantir que le comportement de leurs systèmes de commande est bien celui requis, et ne contient pas d'éléments nuisibles à la sûreté.

Certains secteurs industriels, comme le transport et les industries critiques (processus chimiques, énergie, industrie pétrolière...) semblent être concernés plus particulièrement par la sûreté de la commande. Mais en fait tous les secteurs (processus de fabrication, bâtiment, banque, ...) considèrent de plus en plus sérieusement la sûreté de leurs systèmes de commande comme une exigence fondamentale.

1.1 Sûreté en ligne et sûreté hors ligne

Un système de commande est concerné à deux titres par la sûreté. D'abord il ne doit pas produire lui-même des événements dangereux. De plus, un système de commande exécute souvent des fonctions liées à la sûreté, en surveillant un processus physique, en détectant des déviations potentiellement dangereuses afin d'entreprendre des actions réactives visant à mettre le système physique dans un état sûr ou simplement pour avertir un opérateur.

Deux catégories de personnes sont donc en fait concernées par la sûreté : les concepteurs du système de commande et ses utilisateurs. Les premiers doivent assurer la sûreté du système lors de son élaboration, les deuxièmes lors de son exploitation. En tenant compte de ces deux préoccupations différentes, les méthodes contribuant à la sûreté peuvent être groupées en deux catégories : sûreté hors-ligne et sûreté en-ligne (voir figure 2) [Fau 01]. Le cycle de vie d'un système de commande comporte en effet globalement deux périodes :

- l'élaboration, qui comprend les phases de spécification, de conception, d'implantation, de test et de validation ;
- l'exploitation, pendant laquelle le système de commande est employé pour commander un processus physique ou est entretenu.

La sûreté hors-ligne a pour but d'éviter (ou de limiter) les fautes du système de commande nuisibles à la sûreté. La sûreté en-ligne a pour objectif de garantir le bon fonctionnement du système physique lors de l'exploitation.

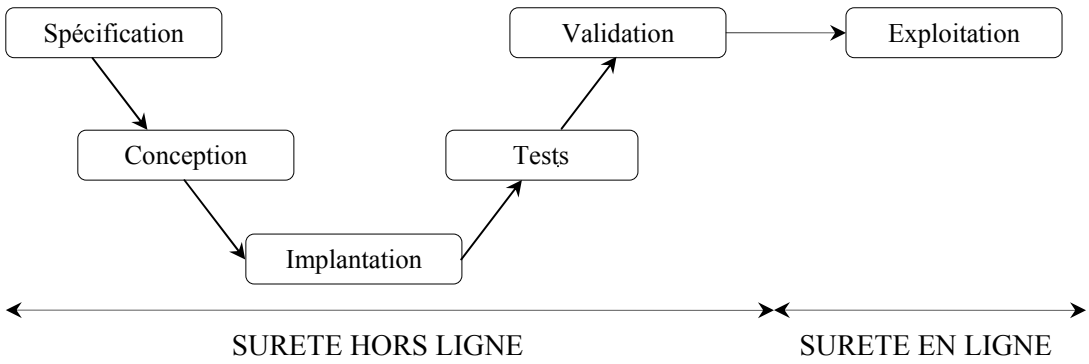


Figure 2 *Sûretés hors-ligne et en-ligne*

Nos travaux se positionnent dans le cadre de la sûreté hors-ligne. Ceci explique que nous allons nous intéresser aux méthodes de conception permettant d'assurer ce type de sûreté.

1.2 Conception de la commande

Dans la pratique industrielle, les systèmes de commande sont malheureusement conçus sans méthode formelle (figure 3).

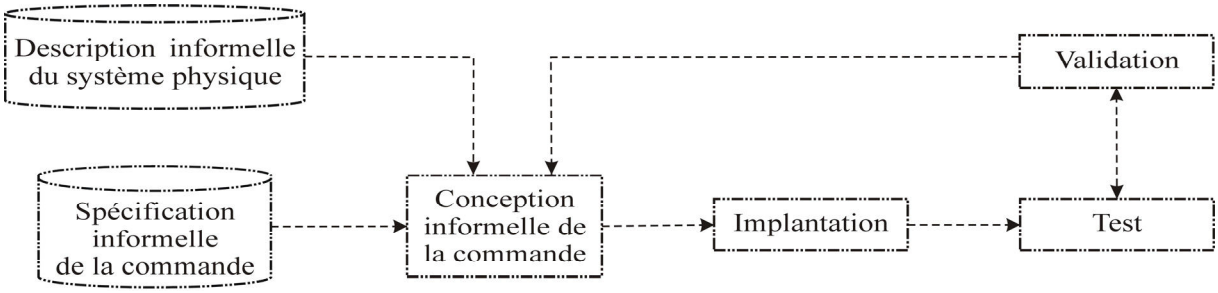


Figure 3 *Conception informelle de la commande*

Les techniques de vérification formelle (figure 4) assurent la sûreté **a posteriori**. En effet, dans cette approche :

1) Un modèle du système de commande est conçu de manière informelle.
2) Ce modèle non sûr a priori est traduit en un modèle formel.
3) Il est alors possible de prouver des propriétés intrinsèques (indépendantes de l'application), telles que stabilité, vivacité, blocage, … (vérification) et extrinsèques (fonction de l'application) sur ce modèle formel.
4) Si l'une quelconque de ces propriétés n'est pas prouvée, il faut reprendre la conception formelle.
5) Lorsque toutes les propriétés sont prouvées, il est possible de passer à l'implantation.

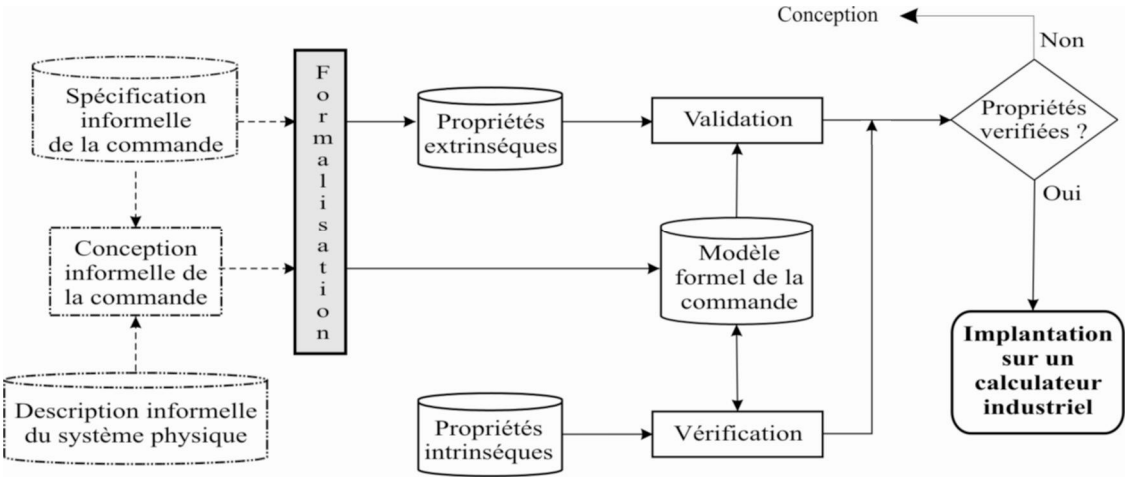


Figure 4 *Vérification et validation formelles de la commande*

Les techniques de synthèse formelle (figure 4) permettent, elles, d'obtenir un système de commande sûr **a priori**. Dans cette approche, le modèle de commande est en effet obtenu directement à partir de modèles formels des spécifications et du système physique à contrôler.

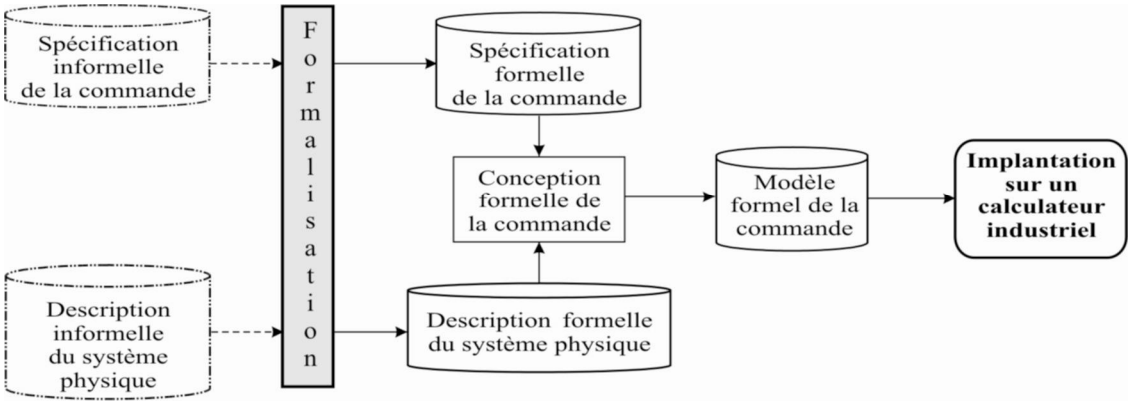


Figure 5 *Synthèse formelle de la commande*

Nos travaux se situant dans le domaine de la synthèse formelle, nous allons à présent faire un état de l'art des travaux scientifiques dans ce domaine.

1.2.1 Synthèse formelle de la commande

Etant donné que nous allons limiter notre travail à un type de système de commande particulier : la commande des Systèmes à Evénements Discrets (SED), nous allons tout d'abord définir les SED. Puis nous aborderons la *théorie de la supervision (ou de la commande supervisée) des SED* qui a été introduite par Ramadge et Wonham [Ram 82], [Won 87], car cette théorie constitue un apport très important au domaine et est celle sur laquelle se base la plupart des travaux.

1.2.1.1 Système à Événements discrets (SED)

Un Système à Evénements Discrets (SED) est un système dynamique dont l'espace d'états est discret et tel que les changements d'état se produisent sur occurrences d'événements [Cas 99]. Ces systèmes permettent de modéliser des systèmes de production manufacturière, de logistique, le trafic de véhicules, les réseaux de communications, etc.

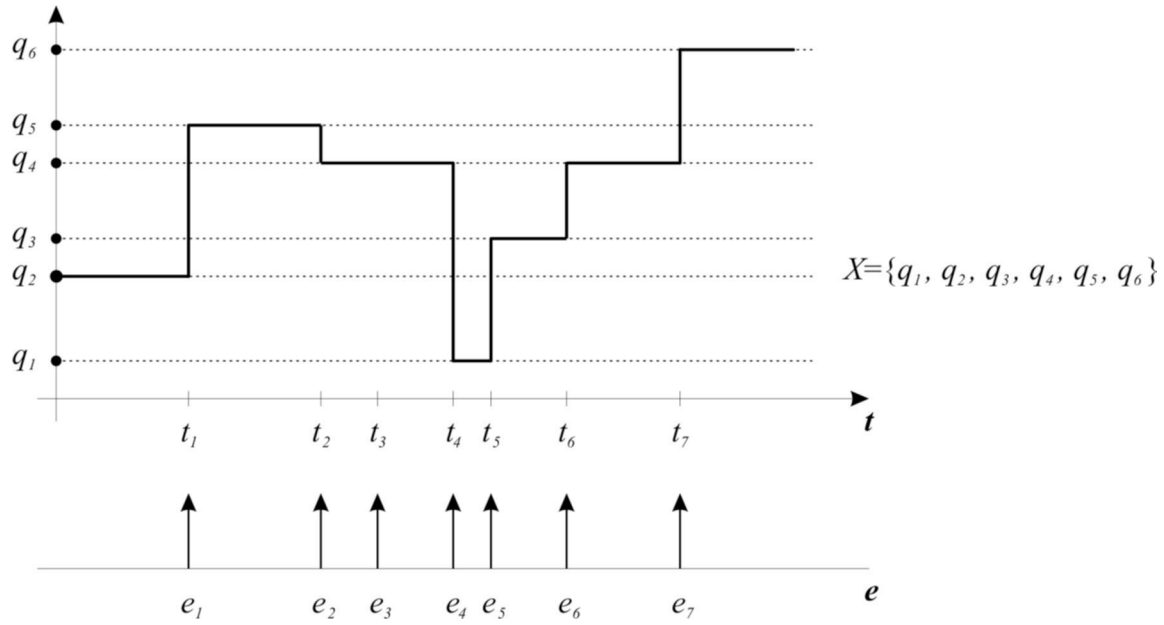


Figure 6 *Modélisation du comportement dynamique d'un SED dans le temps*

1.2.1.2 Modélisation temporelle

Le comportement dynamique d'un SED peut être modélisé avec un temps continu comme il est montré dans la figure 6. Les états discrets sont représentés par q_i et les événements par e_j.
Les SED qui ne considèrent que l'ordre des événements sont appelés des modèles à temps logique. Les modèles qui utilisent des intervalles de temps valués sont appelés des modèles à temps physique.

1.2.1.3 Modélisation par automates

Un automate est un modèle qui permet de représenter le comportement d'un SED à l'aide d'états et de transitions. La figure 7a représente l'automate à états correspondant à la figure 6. Les transitions sont étiquetées par des événements qui représentent les conditions de transition. Plusieurs modèles existent dans la littérature [Hop 79], [Ber 91], [Wol 91], [Cha 96].

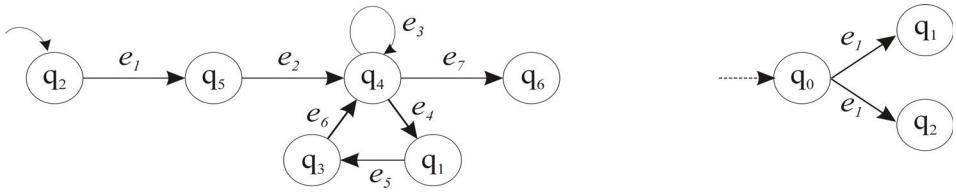


Figure 7 (a) *L'automate à états déterministe de la figure 6* **(b)** *Un automate non déterministe*

1.2.1.4 Automate déterministe

La définition formelle d'un automate déterministe G est : G est un quintuplet G = (Q, Σ, δ, q_0, Q_m) où Q est l'ensemble des états; Σ est l'alphabet d'entrée; δ est la fonction de transition d'états de Q x Σ dans Q ($\delta : \Sigma \times Q \rightarrow Q$) qui associe un état d'arrivée à un état de départ et à un symbole d'entrée; q_0 est l'état initial et, Q_m est l'ensemble des états finals où $Q_m \subseteq Q$. L'automate est déterministe parce que δ est une fonction dans Σ x Q. En d'autres termes, connaissant l'état courant et le symbole d'entrée de l'automate, on peut déterminer sans équivoque le prochain état de l'automate. La figure 7a montre un automate déterministe ; la figure 7b montre un contre exemple : un automate non déterministe.

1.2.1.5 Accepteur [Cha 96]

Un accepteur A est un quadruplet : (Q, Σ, δ, q_0) où Q est l'ensemble des états; Σ est l'alphabet d'entrée; δ est la fonction de transition d'états de Q x Σ dans Q qui associe un état d'arrivée à un état de départ et à un symbole d'entrée; q_0 est l'état initial. On peut remarquer que l'accepteur ne comporte pas d'états finaux.

1.2.1.6 Composition des automates

Plusieurs types de compositions existent tels que la composition libre ou cartésienne, composition synchrone et composition asynchrone. Chacune de ces compositions spécifie la façon selon laquelle les états et les événements des modules G_i sont pris en compte pour décrire le comportement global. Nous allons présenter ci-après la composition synchrone qui est utilisée par la suite.

Composition synchrone :

Si G_1, G_2 et G sont des automates à états.

Si Σ_1, Σ_2 et Σ représentent l'ensemble des événements de G_1, G_2 et G,

Si Q_1, Q_2 et Q représente l'ensemble des états de G_1, G_2 et G respectivement.

La composition synchrone de $G = G_1 \|_s G_2$ est alors définie comme suit :

- *Evénements :* $\Sigma = \Sigma_1 \cup \Sigma_2$.
- *Etats :* $Q = Q_1 \times Q_2$ où l'état initial q initial = (q_1 initial, q_2 initial).
- *Les transitions* de G sont définies comme suit :

Pour $a \in \Sigma_1 \cap \Sigma_2$:

si $(q_1 , a) \rightarrow q_1'$ et $(q_2 , a) \rightarrow q_2'$ alors $\delta : ((q_1 , q_2) , a) \rightarrow (q_1' , q_2')$

Pour a $\notin \Sigma_1 \cap \Sigma_2$:

A) Si a $\in \Sigma_1$ et $(q_1 , a) \rightarrow q_1$' alors $\delta : ((q_1 , q_2) , a) \rightarrow (q_1' , q_2)$

B) Si a $\in \Sigma_2$ et $(q_2 , a) \rightarrow q_2$' alors $\delta : ((q_1 , q_2) , a) \rightarrow ((q_1 , q_2')$

Notons que dans cette composition la simultanéité des événements n'est pas prise en compte.

1.2.1.7 Modélisation d'un SED par langage.

Le comportement d'un SED peut être aussi modélisé par un langage. La théorie des langages [Hop 79] introduit les concepts suivants.

L'alphabet (Σ) est un ensemble fini de symboles ou de lettres (β, δ, α, λ).

Un mot (chaîne ou séquence) est une suite finie de symboles d'un alphabet (par exemple $\beta\alpha\beta\delta\lambda$). Le mot vide est représenté par ε.

Un langage est un ensemble de mots construit sur le même alphabet Σ.

La fermeture itérative (*) est la représentation d'un mot répété un nombre indéfini de fois : 0, 1, 2, …, n. Celle-ci représente une boucle dans un automate. Par exemple, dans la figure 7a, (e_3)* et $(e_4\ e_5\ e_6)$* est la représentation cyclique du mot (e_3) et $(e_4\ e_5\ e_6)$ respectivement, répété n fois (n=0, 1, 2, …, etc.). Note : la chaîne avec n=0, est le mot vide : $(\alpha\lambda\mu)^0 = \varepsilon$.

Les concepts de la théorie des langages permettent de représenter le comportement d'un SED. En effet, l'ensemble des mots décrit le comportement du système. La figure 8 montre la correspondance entre modélisation par automate et modélisation par langage.

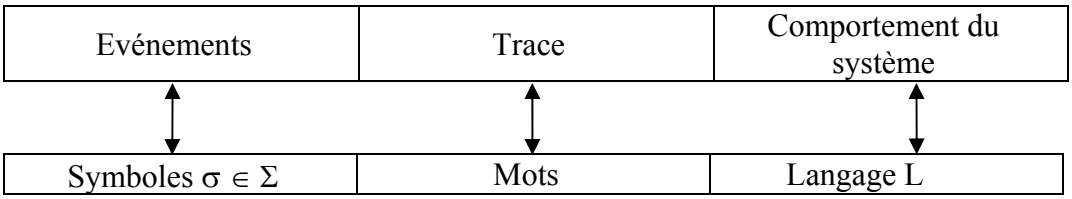


Figure 8 *Relation entre un SED et sa représentation en théorie des langages*

1.2.2 La théorie de la supervision (ou de la commande supervisée)

Dans les années 80, Ramadge et Wonham [Ram 82], [Won 87] ont défini une méthode de conception formelle appelé "*théorie de la supervision des systèmes à événements discrets*". Cette théorie constitue un apport indéniable à la conception de la commande de SED car la plupart de travaux dans le domaine sont dérivés de cette approche.

Cette théorie repose sur une modélisation des SED basée sur des langages formels. Le système à contrôler est considéré comme un générateur spontané d'événements nommé procédé. Son évolution est décrite par un ensemble de séquences d'événements qui forme un

langage. Seuls les SED à temps logique où les événements sont asynchrones sont considérés. La théorie de la commande supervisée repose sur l'idée du contrôle d'un procédé en boucle fermée avec un superviseur, ce superviseur étant un SED qui permettra la modification, en avance, du fonctionnement du procédé en acceptant ou interdisant certains événements afin de respecter la spécification de fonctionnement. Ainsi des techniques [Won 87], [Kum 91] ont été développées afin de synthétiser systématiquement un tel superviseur. Ces travaux sont basés sur le concept de contrôlabilité défini plus bas (1.2.2.2) qui est le concept clef de la théorie. Le but de ces techniques est de garantir la spécification de fonctionnement imposée. Le résultat est considéré comme optimal puisque le fonctionnement obtenu est le plus large possible, nommé fonctionnement suprême contrôlable. Ces deux derniers points constituent l'apport fondamental de la théorie.

1.2.2.1 SED Contrôlable

Un SED représenté par le générateur G équipé avec un ensemble d'entrées de contrôle Γ, est appelé un SED Contrôlable G_c (SEDC). Dans ce qui suit, Γ est fixé. Ainsi, un SEDC est un SED qui génère spontanément les événements Σ. Ces événements sont divisés en deux groupes : événements contrôlables Σ_c et incontrôlables Σ_u où $\Sigma = \Sigma_c \cup \Sigma_u$ (voir figure 9a). En effet, pour contrôler un SED il est supposé qu'il existe des événements qui peuvent être interdits. De telles interdictions ont pour but de prévenir des occurrences d'événements à certains instants afin de respecter une spécification de fonctionnement (voir figure 9b).

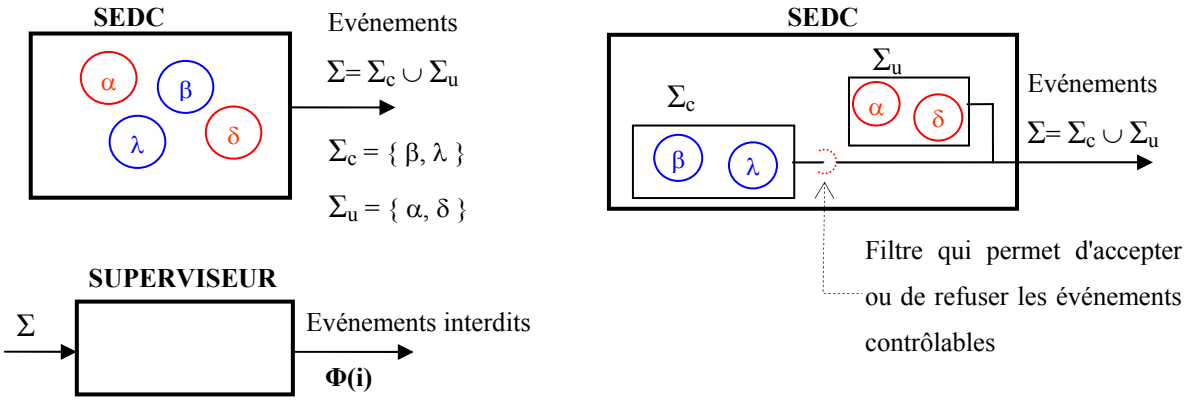


Figure 9 (a) *Illustration SEDC* **(b)** *Représentation d'un SEDC avec la notion de filtrage des événements contrôlables* Σ_c **(c)** *Illustration d'un Superviseur*

1.2.2.2 Contrôlabilité

Un langage K (spécification) est contrôlable par rapport au langage L (SEDC), lorsque un événement Σ_u possible entraîne une évolution vers un état appartenant également à K :

$K\,\Sigma_u \cap L \subseteq K$.

1.2.2.3 Superviseur

Le superviseur peut être perçu (figure 9c) comme un système qui génère à l'instant logique i une liste d'événements interdits Φ(i) inclus dans Σ_c Il est possible de percevoir de façon équivalente le superviseur comme le générateur d'événements acceptés. En effet, la liste d'événements acceptés sera alors Σ_c - Φ(i).

La définition formelle du superviseur [Cha 96] est la suivante :

> Le superviseur S, est une machine de Moore, définie par le 6-uplet $S = (V, \Sigma, \xi, v_0, 2^{\Sigma c}, \theta)$ où V est un ensemble fini d'états; Σ est l'alphabet d'entrée; $\xi : VxS \to V$ est la fonction de transition d'états; v_0 est l'état initial; $2^{\Sigma c}$ est l'alphabet de sortie; et $\theta : V \to 2^{\Sigma c}$ est la fonction d'affectation de sortie.

Ainsi, le superviseur peut être perçu comme une machine à états qui évolue conformément à une modification de son entrée (sur l'occurrence d'un événement de Σ). Pour chaque état v, le superviseur S fournit en sortie une liste d'événements interdits. Rappelons que seuls les événements contrôlables peuvent être interdits, le superviseur n'a aucun droit d'interdire un événement incontrôlable Σ_u. Ainsi chaque sortie de S est un élément de $2^{\Sigma c}$.

1.2.2.4 Supervision en boucle fermée

La théorie considère le couplage d'un SEDC avec un superviseur (voir figure 10). Le but de ce couplage est d'inhiber le comportement du SEDC afin de respecter la spécification de fonctionnement. Le SEDC reçoit en entrée un ensemble d'événements à interdire Φ provenant du superviseur. Le superviseur génère donc la liste d'événements interdits Φ et il reçoit en entrée l'alphabet $\Sigma=(\Sigma_c \cup \Sigma_u)$.

Notons que le rôle du superviseur en boucle fermée est limité à l'interdiction d'occurrences d'événements dans le SEDC car le but de la supervision est d'interdire le fonctionnement non désiré. Ainsi, le superviseur doit être en avance par rapport au SEDC pour accomplir son rôle de superviseur. La figure 10 montre comment le superviseur doit envoyer l'information d'interdiction de l'événement Σ_c qui a été généré par SEDC avant qu'elle puisse être prise en compte comme sortie (Σ) du SEDC.

Le superviseur ne peut pas en aucun cas forcer des événements à se produire. En conséquence, il s'ensuit que le superviseur ne peut que restreindre le fonctionnement du SEDC. Le rôle du superviseur est donc permissif [Ram 87a], [Pet 81]. De plus,

fondamentalement, l'observation du SEDC par le superviseur est asynchrone, c'est-à-dire, non cadencée par un horloge.

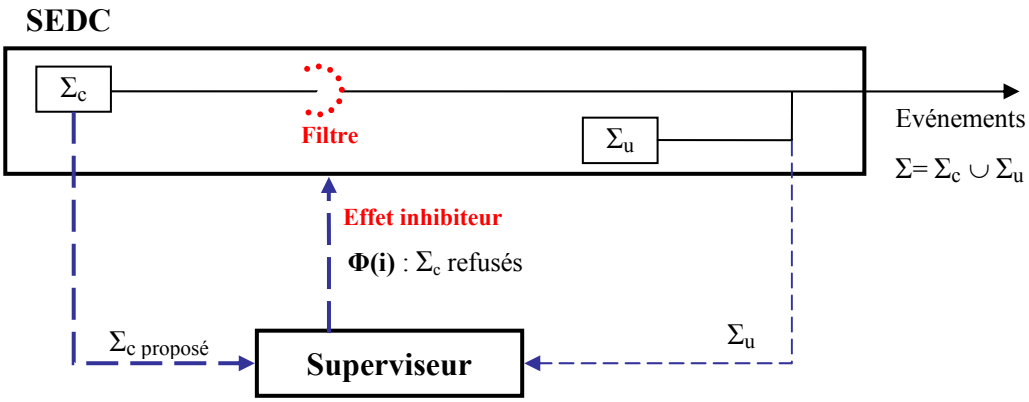


Figure 10 *Illustration du couplage du superviseur et du SEDC en boucle fermée*

1.2.2.5 La synthèse du superviseur

Le principe de la synthèse est le suivant : partant d'un SEDC décrit par un langage L et d'une spécification de fonctionnement donnée par le langage K, le but est de trouver le langage (le plus permissif possible) d'un superviseur qui garantisse que L soit contrôlable par rapport à K.

Intuitivement, la méthode de synthèse d'un tel superviseur (voir figure 11a) consiste à calculer, d'abord, le langage du système formé par le couplage du SEDC (G) et du superviseur (S) par une composition synchrone $L(S/G) = S \|_s G$. Ensuite, on vérifie que L(S) soit contrôlable par rapport à L(S/G).

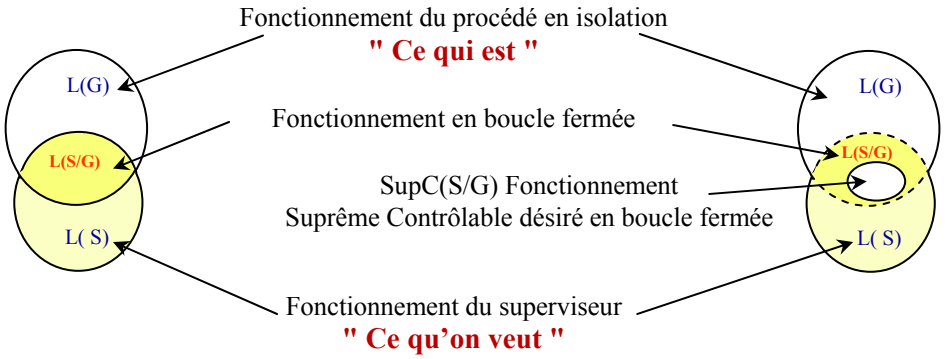


Figure 11 a) *Comportement en boucle fermée* **b)** *Fonctionnement suprême contrôlable*

Si ce n'est pas le cas, on détermine, à l'aide par exemple de l'algorithme de [Kum 91], le langage qui garantisse le fonctionnement sûr et qui soit le plus permissif possible (suprême) SupC(S/G).

1.2.3 Analyse de la théorie de la commande supervisée

De très nombreux travaux ont été réalisés à partir des résultats de Ramadge et Wonham ; ceci met en évidence l'importance de leur apport. Cependant, malgré la richesse de ces résultats, l'application de cette théorie reste encore extrêmement limitée dans l'industrie [Ndj 99]. Afin de comprendre les raisons de cet état de fait, nous analysons ci-après ses limitations :

A) Limitation due à l'explosion combinatoire

B) Limitation due à la construction du SupC(S/G)

C) Limitation due à la modélisation initiale

D) Limitation due à la conception permissive

E) Limitation d'implantation du superviseur

F) Limitation due à la difficulté d'obtenir les modèles de départ

A) Limitation due à l'explosion combinatoire

Pour des modèles non triviaux, l'explosion combinatoire lors de la composition $G||_sS$ rend souvent impossible la synthèse.

Cette limitation a induit la recherche de solutions particulières : supervision modulaire [Ram 87b], [Won 88], [Won 98], [Que 02], supervision sous observation partielle [Ram 86], [Won 94], supervision hiérarchique [Zho 90], [Won 94], [Goh 98], supervision décentralisée [Lin 88], [Han 97]. Ces extensions introduisent d'ailleurs parfois des hypothèses supplémentaires (asynchronisme entre événements de deux modules par exemple) et ne conduisent pas toujours à une solution optimale.

Nous pensons que, pour des SED de grande taille, la modélisation modulaire, hiérarchique, décentralisée est obligatoire, et que ces extensions constituent des apports importants. Cependant, elles n'éliminent pas complètement le problème de l'explosion combinatoire pour des systèmes réels.

B) Limitation due à la construction du SupC(S/G)

Lors de la construction de SupC(S/G), certaines propriétés indispensables telles que l'absence de blocage, l'accessibilité d'un état, peuvent ne pas être respectées. Afin de prévenir ces effets indésirables, il importe de vérifier que le modèle obtenu respecte ces propriétés, ce qui accroît la complexité du calcul.

C) Limitation due à la modélisation initiale

La figure 10, qui fait apparaître un superviseur interdisant certains événements contrôlables d'un SEDC, est différente du modèle classique de commande en boucle fermée (figure 12), qui comprend un procédé à contrôler et un contrôleur.

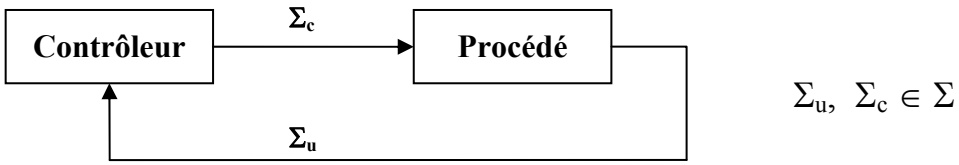


Figure 12 *Système de contrôle- commande*

L'application de la théorie de la supervision est donc confrontée à l'interprétation du modèle de la figure 10 par rapport au modèle de la figure 12.

[Cha 96] a établi un lien entre ces deux interprétations en proposant l'interprétation suivante (figure 13 a) :

1) Le SEDC –nommé procédé étendu– est modélisé par deux SED : un procédé et un contrôleur.
2) Le contrôleur est le générateur de Σ_c, dans le SEDC, et intègre le filtre d'événements.

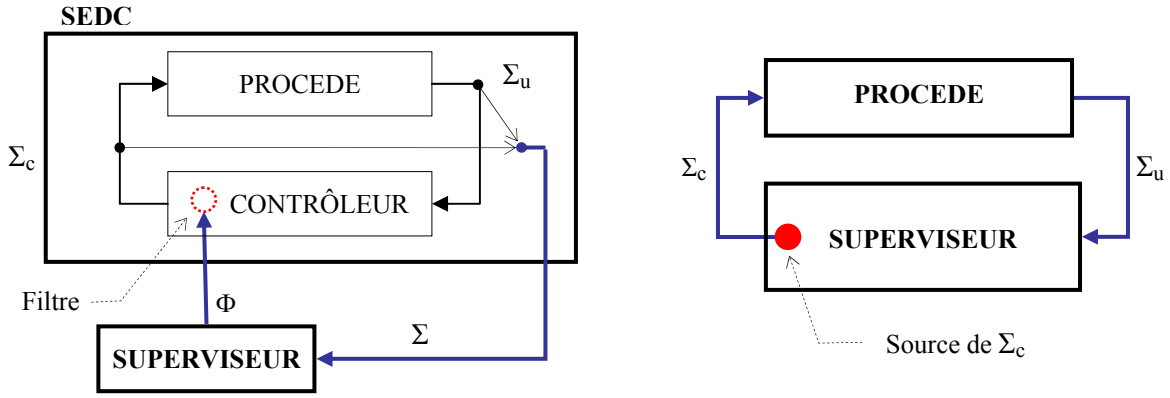


Figure 13 a) *Interprétation de* [Cha_96] **b)** *Schéma du contrôle par supervision par événement forcé selon* [Bal_93]

De plus, ce travail présente un algorithme de liaison entre un modèle Grafcet et le modèle du superviseur synthétisé. Ceci facilite l'applicabilité de la méthode. [Kat 04] a repris ce travail en enlevant l'hypothèse d'une spécification contrôlable. Ces travaux sont un apport vers l'application au niveau industriel.

D) Limitation due à la conception permissive

La théorie initiale ne permet que la conception d'une commande permissive car elle ne fait qu'interdire l'occurrence d'événements Σ_c générés dans le SEDC. Or, de nombreux problèmes de commande imposent de forcer, et pas seulement d'interdire, des événements. Ceci a conduit au développement d'une autre approche basée sur le concept d'événements forcés ([Bra 89], [Bal 92], [Bal 93], [Li 91]). La figure 13b illustre cette approche, connue comme contrôle de type entrées/sorties. Le SEDC ne génère plus Σ_c et devient le procédé ; la notion de filtre disparaît et le rôle du superviseur est de fournir Σ_c.

E) Limitation d'implantation du superviseur

Il n'existe que peu de travaux qui s'intéressent à l'implantation d'un superviseur. Nous pouvons citer [Bal 93], [Lau 96], [Gou 02], [Que 02], [Kat 04], [Taj 05]. Ces travaux bien qu'intéressants présentent cependant des limitations sur l'implantation.

D'autre part, dans les travaux de [Fab 98], [Hel 02], [Gou 04], il est proposé un équivalent en "*Ladder Diagram*" du superviseur, sous réserve que certaines hypothèses soient respectées. Du point de vue théorique, l'applicabilité de l'approche par événements forcés est alors possible.

F) Limitation due à la difficulté d'obtenir les modèles de départ

Les modèles formels du procédé et de la spécification, supposés existants dans la théorie, conditionnent fortement le résultat de la synthèse. L'obtention de ces modèles joue donc un rôle crucial dans les techniques de synthèse et peut conduire à une solution rapide et simple ou, dans le pire de cas, à l'impossibilité de résultat. [Gou 04] présente un tableau qui résume ce problème et qui met en évidence, en plus, que l'absence de résultat par synthèse ne garantit pas la non-existence d'une solution.

Nous avons constaté ceci, d'ailleurs, en participant au groupe de travail COSED sur la base d'un cas d'étude simple. A partir de la spécification de fonctionnement en langage naturel d'une commande simple d'un module d'une machine d'assemblage réelle[1], avec 6 entrées et 4 sorties à commander, des modèles ont été développés par plusieurs laboratoires [Gaf 03], [Gou 03], [Phi 03], [Rou 03a], [Taj 03]. Ces modèles étaient de taille et de granularité très différentes, même si la taille du cas traité était réduite, ce qui a conduit à des solutions de

[1] La machine d'assemblage est installée au laboratoire d'automatique du Département de Génie Mécanique de l'ENS Cachan.

complexités très variables. Il est possible de faire le même constat, à travers l'étude de [Pin 99], dans laquelle une étude comparative, sur deux exemples, de la méthode de synthèse en modélisant par automates, par RdP, par TTM (Timed Transition Models) et par NCES (Net Conditions Events) a été faite.

1.2.4 Conclusions

La théorie de la supervision constitue un apport fondamental pour la synthèse des SED. Un nombre extrêmement important de travaux sont basés sur cette théorie, en utilisant des modélisations par automates, réseaux de Petri (travaux de [Gha 02], [Did 06, notamment), TTM ([Ost 89]), NCES ([Han 95], [Rau 95]). Cependant cette théorie ***est une méthode de conception formelle qui ne s'intéresse pas à l'implantation de la commande***❶. En effet, "les procédures de synthèse fournissent un modèle d'un superviseur abstrait qui ne peut pas être implanté en l'état sur un système réel" [Ndj 99]. Ceci réduit son intérêt pratique. D'autre part, elle s'appuie sur des ***modèles abstraits pour un concepteur industriel***❷.

En troisième lieu, il a été constaté que ***la modélisation du procédé et de la spécification*** ❸ joue un rôle crucial pour l'obtention des résultats. Un concepteur est alors confronté au problème de savoir comment modéliser son procédé et ses besoins sans provoquer : 1) un temps de calcul trop grand, 2) l'échec du calcul (aucune solution), 3) le non accomplissement d'une propriété (vivacité, blocage, états marqués, etc.). Le point 1 est relatif aux moyens du calcul ; par contre les points 2 et 3 imposent ***une communication avec le concepteur***❹ afin de l'aider à détecter les incohérences dans les modèles initiaux. L'apport de [Taj 05] est notable à ce sujet cependant ce travail suppose que le concepteur soit expert. En effet, ***les résultats d'un échec lors de la synthèse, s'il y en a, ne sont pas toujours évidents pour un concepteur industriel***❺.

Enfin, en dépit des travaux sur la supervision modulaire, l'***explosion combinatoire***❻ demeure une limitation forte pour l'application au niveau industriel de la théorie.

Ainsi, en tenant en compte des limitations : ❶, ❷, ❸, ❹, ❺ et ❻, il nous est apparu nécessaire de chercher une autre alternative que nous présentons ci-après.

1.3 Approche proposée

Nous présentons dans la suite de ce document les résultats de nos travaux qui visent à proposer une méthode alternative à l'approche de commande supervisée. Cette méthode de

synthèse formelle est conçue pour *produire un système de commande sûr a priori* et considère également *l'implantation de cette commande.*

La figure 14 montre le principe de cette méthode de synthèse qui s'appuie un *cadre formel algébrique* : l'algèbre $\mathbb{I}$. La spécification formelle de la commande est obtenue à partir des modèles informels des spécifications et du système physique en langage naturel (LN). A partir des spécifications formelles, la phase de conception permet d'obtenir les fonctions de contrôle des sorties du système de commande. Dans cette phase nous vérifions que les spécifications formelles respectent certaines propriétés, notamment de *cohérence* et de *complétude*, qui sont nécessaires pour obtenir une fonction de contrôle solution. Ceci est réalisé à l'aide d'algorithmes développés spécifiquement et grâce à la communication avec le concepteur.

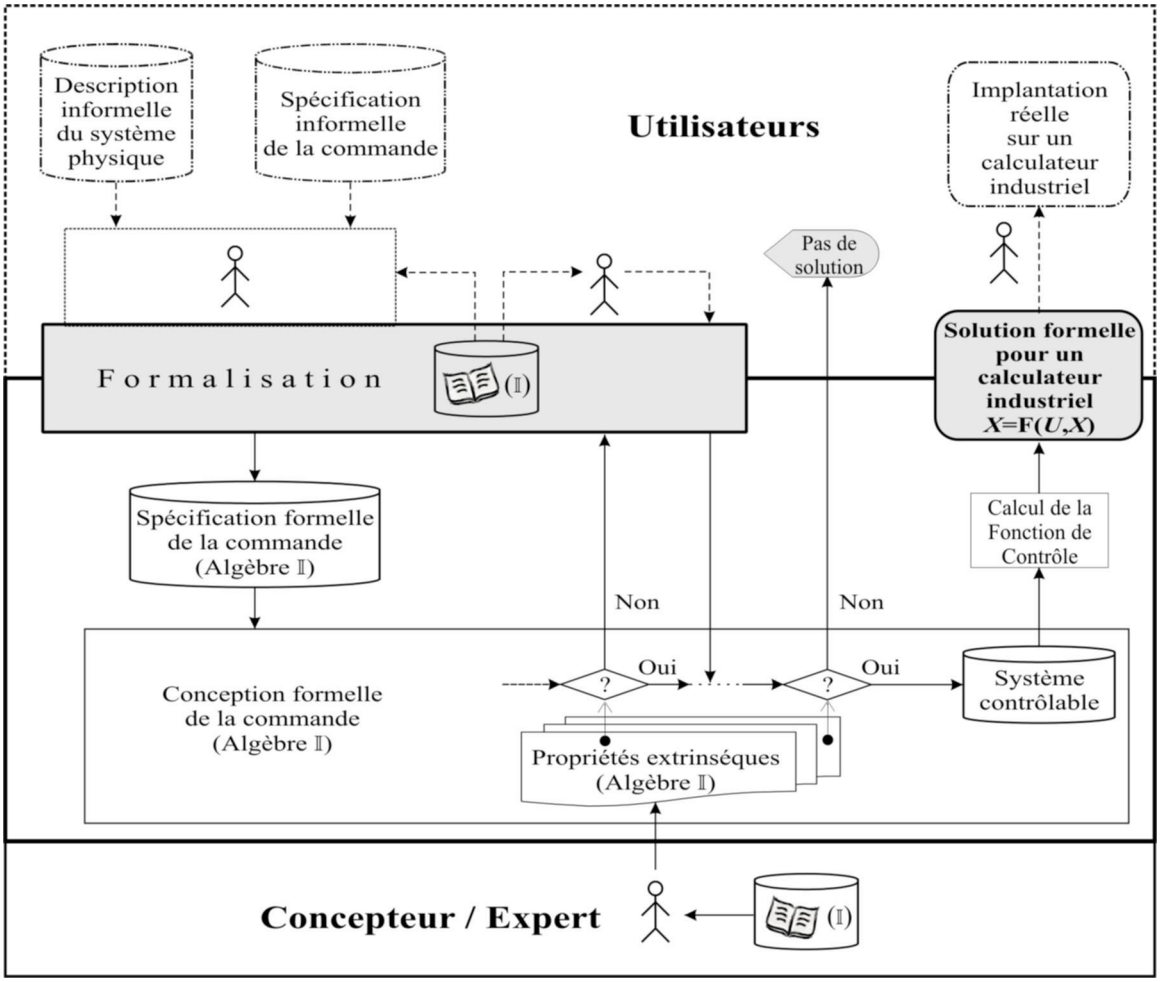


Figure 14 *Schéma de la méthode d'élaboration sûre de la commande proposée avec l'algèbre* $\mathbb{I}$

Pour un système de commande *m* entrées $u_1, u_2, \quad , u_m$ et à n sorties $X_1, X_2, \quad , X_n$, les n fonctions de contrôle $F_1, F_2, \quad , F_n$ peuvent être implantées de façon à être exécutées en

parallèle ou séquentiellement. Ces deux cas seront considérés dans notre étude, en nous limitant à un système centralisé (implantation dans un seul calculateur ou API).

La méthode proposée considère une modélisation initiale en LN qui puisse être formalisée et elle considère l'implantation du résultat obtenu (fonction du contrôle) sur un calculateur industriel. Ceux-ci avec l'objectif de surmonter les limitations C, E et F (voir 1.2.3). De plus, nous prendrons en compte des spécifications d'interdiction et d'imposition des événements afin de surmonter la limitation D.

En plus, dans notre approche nous ne considérons pas la construction d'un superviseur (limitation B). Or, nous proposerons des expressions pour vérifier des propriétés du comportement global mais a vérifier directement sur les fonctions des sorties afin d'éviter l'explosion combinatoire (limitation A). De plus, l'algèbre $\mathbb{I}$ nous permettra d'utiliser un seul cadre formel tout au long de la méthode, dès la formalisation jusqu'à l'implantation. Nous éviterons ainsi l'introduction des hypothèses.

Organisation du mémoire

Les deux chapitres suivants présentent le cadre formel algébrique sur lequel s'appuie notre méthode de synthèse. Le chapitre 4 est consacré à l'obtention de spécifications formelles à partir d'énoncés en langage naturel (LN). Dans le chapitre 5, nous présentons une méthode de résolution de système d'équations qui sert de base à la synthèse de fonctions de contrôle. Enfin, cette méthode de synthèse est détaillée dans le chapitre 6 tandis que les chapitres 7 et 8 l'illustrent à l'aide de trois exemples.

Chapitre 2

Les bases de l'algèbre $\mathbb{I}$, qui est le cadre mathématique de nos travaux, sont présentées dans ce chapitre. Cette algèbre est une algèbre de Boole pour les signaux binaires.

Les opérations de base (ET, OU et NON) et la relation d'égalité sont présentées dans un premier temps. Ensuite, nous introduisons la relation d'ordre partiel "$\leq$", appelée inclusion, qui est un point clef pour la résolution de systèmes d'équations développée au chapitre 5.

2. Bases d'une algèbre pour les signaux binaires

2.1 Introduction

Le cadre mathématique présenté dans ce document (appelé algèbre $\mathbb{I}$) est conçu pour l'étude des systèmes logiques, c'est-à-dire les systèmes pour lesquels les variables ne peuvent prendre que les deux seules valeurs 0 ou 1.

De plus, dans le cadre mathématique présenté ci-dessous, le temps est considéré comme une grandeur continue représentée par une variable réelle. Cette modélisation du temps, plus conforme à la perception du temps dans le monde physique, permet de s'affranchir des différents problèmes d'échantillonnage tout en simplifiant la définition mathématique des primitives de cette algèbre. Ainsi, les primitives du cadre formel (opérations et relations), présentées dans ce chapitre, permettent de décrire sous forme d'équations algébriques des comportements statiques ou dynamiques de systèmes physiques logiques.

Dans la section suivante, l'algèbre $\mathbb{I}$, ses opérations de base : ET, OU, NON et deux relations : égalité et inclusion ($\leq$) sont présentées. Cette dernière relation, qui est une relation d'ordre partiel, est primordiale pour la résolution des systèmes d'équations développée au chapitre 5.

2.2 Structure d'Algèbre de Boole

Il existe plusieurs définitions différentes dans la littérature pour une algèbre de Boole. Pour éviter toute ambiguïté due à l'utilisation de définitions élaborées par des auteurs ayant des points de vue différents, les définitions de chacun des concepts mathématiques utilisés dans ce document sont toutes extraites d'un seul document. Il s'agit de [Gri 00] :

Discrete and Combinatorial Mathematics: An Applied Introduction
R. P. GRIMALDI
New-York, Addison-Wesley Editions, 4ième edition
ISBN : 0-201-19912-2.

Nous ferons principalement référence au chapitre 15.4 intitulé "The structure of a Boolean Algebra".

2.2.1 Définition mathématique d'un signal binaire

La forme générale d'un signal binaire est décrite sur la figure 15. Elle correspond à une fonction de $\mathbb{R}^{+*}$ dans $\mathbb{B} = \{0,1\}$ continue par morceaux, pouvant admettre en certains points une double discontinuité. C'est le cas des dates t_2 et t_6 pour le signal représenté sur la figure 15. Pour les dates telles que t_1 et t_3, le signal de cette figure est continu à droite.

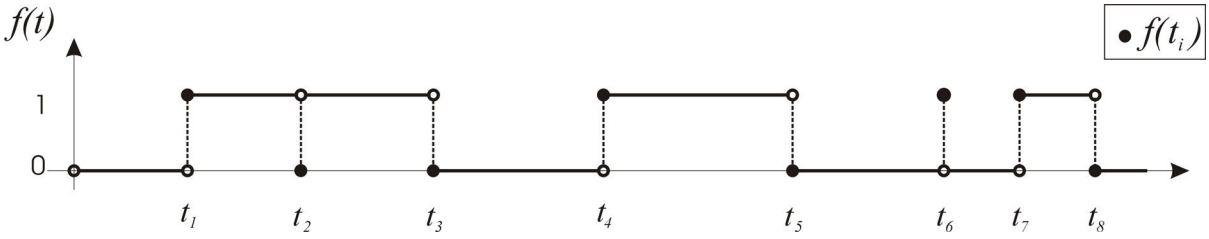


Figure 15 *Représentation graphique d'un signal binaire* f

Soit $\mathbb{I}$ l'ensemble de ces fonctions. Cet ensemble $\mathbb{I}$ peut être défini mathématiquement comme suit (définition 1). La continuité par morceaux des éléments de $\mathbb{I}$ est matérialisée par l'existence pour toute date t de $\mathbb{R}^{+*}$ d'un intervalle ouvert de longueur aussi petite que l'on veut mais non nulle sur lequel toutes les valeurs de la fonction f sont identiques.

Définition 1 : Signal binaire

$$\mathbb{I} = \{f : \mathbb{R}^{+*} \to \mathbb{B} \mid \forall t \in \mathbb{R}^{+*} : (\exists \varepsilon_t > 0 : (\forall (\varepsilon_1, \varepsilon_2) \in]0, \varepsilon_t[^2, f(t-\varepsilon_1) = f(t-\varepsilon_2)))\}$$

2.2.2 Éléments caractéristiques

Parmi tous les éléments de $\mathbb{I}$, deux fonctions de $\mathbb{R}^{+*}$ dans $\mathbb{B} = \{0,1\}$ auront un rôle prépondérant au sein de cette algèbre. Ces fonctions, notées 1* et 0*, sont définies ainsi :

Définition 2 : Signal 1*

$$1^* : \quad \mathbb{R}^{+*} \to \mathbb{B}, \quad t \mapsto 1$$

Définition 3 : Signal 0*

$$0^* : \quad \mathbb{R}^{+*} \to \mathbb{B}^*, \quad t \mapsto 0$$

Les fonctions 1* et 0* sont les deux seules fonctions de $\mathbb{I}$ pour lesquelles la valeur reste constante sur $\mathbb{R}^{+*}$.

2.2.3 Convention de notation

Pour distinguer les opérations entre les éléments de $\mathbb{I}$ des opérations entre des variables logiques, la convention suivante sera suivie dans ce document :

- Les notations "$\wedge$", "$\vee$" et "$\neg$" seront respectivement utilisés pour les opérations ET, OU, NON entre des variables logiques.
- Les notations "$\cdot$", "+" et "$\bar{}$" seront utilisées pour les opérations ET, OU, NON entre des signaux binaires.
- Les éléments de $\mathbb{I}$ (fonctions de $\mathbb{R}^{+*}$ dans $\mathbb{B} = \{0,1\}$) seront notés f, g et h.
- Les variables logiques représentant les valeurs prises par ces fonctions à un instant t donné seront notées $f(t)$, $g(t)$ et $h(t)$.
- Nous reprendrons la même convention de priorité que l'algèbre de Boole classique : d'abord la parenthèse puis l'opération ET ensuite l'opération OU. Il sera donc possible de supprimer certaines parenthèses lorsque celles-ci ne seront pas strictement nécessaires.
- Un système d'équations sera représenté par une accolade ouvrante à gauche "$\{$ ".

Afin de ne pas surcharger inutilement les notations, le symbole " = " sera utilisé soit entre des variables logiques, soit entre des signaux binaires. La définition de la relation d'égalité est donnée par :

Relation	Définition mathématique	Interprétation usuelle	Caractéristiques
Egalité $f = g$	$\forall t \in \mathbb{R}^{+*}, \left(f(t) = g(t)\right)$	Les signaux f et g sont identiques.	Relation symétrique, transitive et réflexive

2.2.4 Opérations de base

Les trois opérations de base suivantes sont nécessaires pour donner à $\mathbb{I}$ une structure d'algèbre de Boole. Elles permettent de composer les signaux binaires entre eux afin d'obtenir de nouveaux signaux binaires.

La définition mathématique de ces opérations est la suivante :

Définition 4 :
Opération **ET**

$$\begin{aligned} \mathbb{I}^2 &\to \mathbb{I} \\ (f,g) &\mapsto (f \cdot g) \end{aligned}$$
$$\text{Où } \forall t \in \mathbb{R}^{+*},\quad (f \cdot g)(t) = f(t) \wedge g(t)$$

Définition 5 :
Opération **OU**

$$\begin{aligned} \mathbb{I}^2 &\to \mathbb{I} \\ (f,g) &\mapsto (f + g) \end{aligned}$$
$$\text{Où } \forall t \in \mathbb{R}^{+*},\quad (f + g)(t) = f(t) \vee g(t)$$

Définition 6 :
Opération **NON**

$$\begin{aligned} \mathbb{I} &\to \mathbb{I} \\ f &\mapsto \bar{f} \end{aligned}$$
$$\text{Où } \forall t \in \mathbb{R}^{+*},\quad \bar{f}(t) = \neg f(t)$$

Ces trois opérations permettent de composer les éléments de $\mathbb{I}$ de manière combinatoire.

- La valeur $(f \cdot g)(t)$ du signal $(f \cdot g)$ à la date t est obtenue par un ET logique entre deux variables logiques. Il s'agit des valeurs $f(t)$ et $g(t)$ des signaux f et g *à cette même date* t.
- La valeur $(f + g)(t)$ du signal $(f + g)$ à la date t est obtenue par un OU logique entre deux variables logiques. Il s'agit des valeurs $f(t)$ et $g(t)$ des signaux f et g *à cette même date* t.
- La valeur $\overline{f}(t)$ du signal $\overline{f}$ à la date t est obtenue en complémentant la valeur d'une variable logique. Il s'agit de la valeur complémentaire de $f(t)$ du signal f *à cette même date* t.

2.2.5 Structure d'Algèbre de Boole

2.2.5.1 Définition

D'après [Gri 00] page 687, pour démontrer que ($\mathbb{I}$, $\cdot$, $+$, $^{-}$, 1^*, 0^*) a une structure d'Algèbre de Boole, il suffit de vérifier que les neuf conditions suivantes sont satisfaites pour tout élément f, g et h de $\mathbb{I}$.

Commutativité :

$$f \cdot g = g \cdot f \qquad [2.1]$$

$$f + g = g + f \qquad [2.2]$$

Distributivité :

$$f \cdot (g + h) = (f \cdot g) + (f \cdot h) \qquad [2.3]$$

$$f + (g \cdot h) = (f + g) \cdot (f + h) \qquad [2.4]$$

Identité (éléments neutres) :

$$f \cdot 1^* = f \qquad [2.5]$$

$$f + 0^* = f \qquad [2.6]$$

Relations impliquant un signal et son complément :

$$f \cdot \overline{f} = 0^* \qquad [2.7]$$

$$f + \overline{f} = 1^* \qquad [2.8]$$

$$0^* \neq 1^* \qquad [2.9]$$

2.2.5.2 Vérification de la structure d'Algèbre de Boole

Pour démontrer les dernières conditions nous allons utiliser la relation d'égalité donnée en 2.2.3 (cette relation est approfondie à la section 2.3.3) :

Démonstration de l'équation [2.1] : $f \cdot g = g \cdot f$

$$\forall t \in \mathbb{R}^{+*}, \quad (f \cdot g)(t) = f(t) \wedge g(t) = g(t) \wedge f(t) = (g \cdot f)(t)$$ □

Démonstration de l'équation [2.2] : $f + g = g + f$

$$\forall t \in \mathbb{R}^{+*}, \quad (f + g)(t) = f(t) \vee g(t) = g(t) \vee f(t) = (g + f)(t)$$ □

Démonstration de l'équation [2.3] : $f \cdot g = g \cdot f$

$$\begin{aligned} \forall t \in \mathbb{R}^{+*}, \quad \Big(f \cdot (g+h)\Big)(t) &= f(t) \wedge (g+h)(t) = f(t) \wedge \Big(g(t) \vee h(t)\Big) \\ &= \Big(f(t) \wedge g(t)\Big) \vee \Big(f(t) \wedge h(t)\Big) \\ &= (f \cdot g + f \cdot h)(t) \end{aligned}$$ □

Démonstration de l'équation [2.4] : $f + (g \cdot h) = (f + g) \cdot (f + h)$

$$\begin{aligned} \forall t \in \mathbb{R}^{+*}, \quad \Big(f + (g \cdot h)\Big)(t) &= f(t) \vee (g \cdot h)(t) = f(t) \vee \Big(g(t) \wedge h(t)\Big) \\ &= \Big(f(t) \vee g(t)\Big) \wedge \Big(f(t) \vee h(t)\Big) \\ &= \Big((f+g) \cdot (f+h)\Big)(t) \end{aligned}$$ □

Démonstration de l'équation [2.5] : $f \cdot 1^* = f$

$$\forall t \in \mathbb{R}^{+*}, \quad (f \cdot 1^*)(t) = f(t) \wedge 1^*(t) = f(t) \wedge 1 = f(t)$$ □

Démonstration de l'équation [2.6] : $f + 0^* = f$

$$\forall t \in \mathbb{R}^{+*}, \quad (f + 0^*)(t) = f(t) \vee 0^*(t) = f(t) \vee 0 = f(t)$$ □

Démonstration de l'équation [2.7] : $f \cdot \overline{f} = 0^*$

$$\forall t \in \mathbb{R}^{+*}, \quad (f \cdot \overline{f})(t) = f(t) \wedge \overline{f}(t) = f(t) \wedge \neg f(t) = 0 = 0^*(t)$$ □

Démonstration de l'équation [2.8] : $f + \overline{f} = 1^*$

$$\forall t \in \mathbb{R}^{+*}, \quad (f + \overline{f})(t) = f(t) \vee \overline{f}(t) = f(t) \vee \neg f(t) = 1 = 1^*(t)$$ □

Vérification de l'équation [2.9] : $0^* \neq 1^*$

Ces deux signaux binaires sont différents par définition. □

Comme les neuf conditions nécessaires sont respectées, alors ($\mathbb{I}$, $\cdot$, +, $^{-}$, 1^{*}, 0^{*}) a bien une structure d'Algèbre de Boole.

2.2.5.3 Propriétés communes aux algèbres de Boole

Les 11 théorèmes suivants sont des théorèmes communs à toute algèbre de Boole [Gri 00] page 689. Ils découlent des neuf conditions nécessaires pour obtenir une algèbre de Boole. Comme ($\mathbb{I}$, $\cdot$, +, $^{-}$, 1^{*}, 0^{*}) est une Algèbre de Boole, les théorèmes suivants sont donc obtenus pour tout élément f, g et h de $\mathbb{I}$ (voir démonstrations dans l'annexe B) :

Idempotence :

$$f \cdot f = f \qquad [2.10]$$

$$f + f = f \qquad [2.11]$$

Eléments neutres :

$$f \cdot 0^{*} = 0^{*} \qquad [2.12]$$

$$f + 1^{*} = 1^{*} \qquad [2.13]$$

Absorption :

$$f + (f \cdot g) = f \qquad [2.14]$$

$$f \cdot (f + g) = f \qquad [2.15]$$

Associativité :

$$f \cdot (g \cdot h) = (f \cdot g) \cdot h \qquad [2.16]$$

$$f + (g + h) = (f + g) + h \qquad [2.17]$$

Loi de double négation :

$$\overline{\overline{f}} = f \qquad [2.18]$$

Lois de De Morgan :

$$\overline{(f \cdot g)} = \overline{f} + \overline{g} \qquad [2.19]$$

$$\overline{(f + g)} = \overline{f} \cdot \overline{g} \qquad [2.20]$$

Dans la suite des travaux, nous ferons référence à des théorèmes complémentaires à ceux proposés dans [Gri 00] page 689. Ceux-ci seront regroupés dans cette partie du document et démontrés à partir des résultats précédents dans l'annexe B.

$$f + (\overline{f} \cdot g) = f + g \qquad [2.21]$$

$$f \cdot g + \overline{f} \cdot h + g \cdot h = f \cdot g + \overline{f} \cdot h \quad \text{(théorème de la redondance)} \qquad [2.22]$$

$$\overline{1*} = 0* \qquad [2.23]$$

$$\overline{0*} = 1* \qquad [2.24]$$

$$f + g = 0* \quad \Leftrightarrow \quad \begin{cases} f = 0* \\ g = 0* \end{cases} \qquad [2.25]$$

$$f \cdot g = 1* \quad \Leftrightarrow \quad \begin{cases} f = 1* \\ g = 1* \end{cases} \qquad [2.26]$$

2.3 Relations d'égalité et d'inclusion entre des signaux binaires

2.3.1 Définitions des relations d'égalité, de différence et d'inclusion

Le tableau 1 regroupe l'ensemble des relations entre des éléments de $\mathbb{I}$ que nous utilisons.

Relation	Définition mathématique	Interprétation usuelle	Caractéristiques
Egalité $f = g$	$\forall t \in \mathbb{R}^{+*}, \left(f(t) = g(t)\right)$	Les signaux f et g sont identiques.	Relation symétrique, transitive et réflexive
Différence $f \neq g$	$\exists t_i \in \mathbb{R}^{+*}, \left(f(t_i) \neq g(t_i)\right)$	Il existe au moins une date t_i pour laquelle $f(t_i)$ est différent de $g(t_i)$.	Relation seulement symétrique
Inclusion $f \leq g$	$f \cdot g = f$	Pour tout date t où le signal $f(t)$ vaut 1, le signal $g(t)$ vaut également 1.	Relation réflexive, transitive et anti-symétrique

Tableau 1 *Ensemble des relations entre des signaux binaires*

Rappelons qu'une relation R entre deux éléments a et b d'un ensemble S est :

- symétrique si et seulement si : aRb est équivalent à bRa,
- transitive si et seulement si : aRb et bRc implique aRc,
- réflexive si et seulement si : $\forall a \in S,\ aRa$,
- anti-symétrique si et seulement si : aRb et bRa implique $a = b$

2.3.2 Relation d'ordre partiel

La relation "$\leq$" correspond à la relation d'ordre partiel qui peut être définie sur **toute** algèbre de Boole [Gri 00] page 691 : $f \leq g \equiv f \cdot g = f$. Dans le cadre de l'algèbre $\mathbb{I}$, cette relation est notée «Inclusion» car, lorsque deux signaux f et g satisfont $f \leq g$, alors, si la valeur $f(t)$ du signal f est 1, quelle que soit la date t de $\mathbb{R}^{+*}$, alors la valeur $g(t)$ du signal g est également de 1. Sur la figure suivante sont présentés les signaux f et g qui satisfont $f \leq g$.

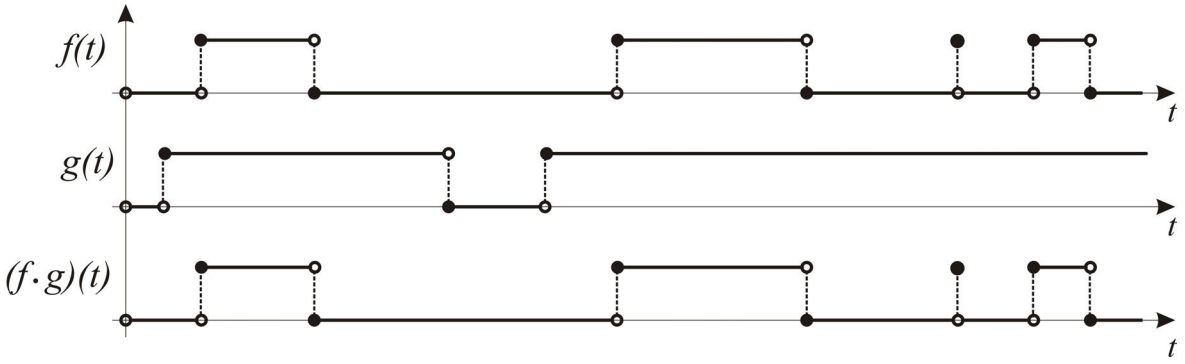


Figure 16 *Exemple de signaux binaires satisfaisant la relation inclusion : $f \leq g \equiv f \cdot g = f$*

2.3.2.1 Caractéristiques de la relation «inclusion»

Les propriétés de réflexivité ([2.27]), de transitivité ([2.28]) et d'anti-symétrie ([2.29]) sont des caractéristiques qui prouvent le fait que "$\leq$" est bien une relation d'ordre partiel.

$$f \leq f \qquad [2.27]$$

$$\begin{cases} f \leq g \\ g \leq h \end{cases} \Rightarrow f \leq h \qquad [2.28]$$

$$\begin{cases} f \leq g \\ g \leq f \end{cases} \Rightarrow f = g \qquad [2.29]$$

Démonstration de la réflexivité de l'inclusion [2.27] : $f \leq f$

Pour tout signal f élément de $\mathbb{I}$ cette propriété est accomplie car l'idempotence [2.10] nous dit que $f \cdot f = f$ et donc cela implique par définition $f \leq f$ □

Démonstration de la transitivité de l'inclusion [2.28] : $\begin{cases} f \leq g \\ g \leq h \end{cases} \Rightarrow f \leq h$

Si $f \leq g$ et $g \leq h$, nous avons par définition $f \cdot g = f$ et $g \cdot h = g$; de sorte que (en utilisant [2.16]), $f \cdot h = (f \cdot g) \cdot h = f \cdot (g \cdot h) = f \cdot g = f$,

par conséquent $f \leq h$ □

Démonstration de l'anti-symétrie de l'inclusion [2.29] : $\begin{cases} f \le g \\ g \le f \end{cases} \Rightarrow f = g$

Si $f \le g$ et $g \le f$, alors $f \cdot g = f$ et $g \cdot f = g$; comme $f \cdot g = g \cdot f$ ([2.1]),
nous avons $f = g$ □

Pour la relation «inclusion», nous avons également les caractéristiques suivantes qui sont vraies pour tout signal f, g de $\mathbb{I}$.

$$f \le 1* \qquad [2.30]$$

$$0* \le f \qquad [2.31]$$

$$f \le (f + g) \qquad [2.32]$$

$$(f \cdot g) \le f \qquad [2.33]$$

Remarques :

Ces propriétés nous indiquent que pour tout signal f et g de $\mathbb{I}$:

- Tout signal (f) est toujours inclus dans le signal 1*, c'est-à-dire $f \le 1*$ est toujours vraie.
- Le signal 0* est toujours inclus dans un signal quelconque (f) de $\mathbb{I}$, c'est-à-dire $0* \le f$ est toujours vraie.
- Un signal (f) est inclus dans un signal composé de ce signal et d'un autre signal quelconque (g) avec l'opération OU ($f + g$), c'est-à-dire $f \le f + g$ est toujours vraie.
- La composition d'un signal (f) et d'un autre signal (g) avec l'opération ET ($f \cdot g$), est inclus dans ce signal (f), c'est-à-dire $f \cdot g \le f$ est toujours vraie.

Les propriétés [2.30], [2.31], [2.32] et [2.33] seront démontrées à partir de la définition $f \le g \equiv f \cdot g = f$

Démonstration de la relation [2.30] : $f \le 1*$

Comme $f \cdot 1* = f$, (en utilisant [2.5])
alors, $f \le 1*$ □

Démonstration de la relation [2.31] : $0* \le f$

Comme $0* \cdot f = 0*$, (en utilisant [2.12])
alors, $0* \le f$ □

Démonstration de la relation [2.32] : $f \le (f + g)$

Comme $f \cdot (f + g) = f$, (en utilisant [2.15])

alors, $f \le (f + g)$ □

Démonstration de la relation [2.33] : $(f \cdot g) \le f$

Comme $(f \cdot g) \cdot f = g \cdot (f \cdot f) = f \cdot g$, (en utilisant [2.1], [2.16]; [2.10], [2.1])

alors, $(f \cdot g) \le f$ □

2.3.2.2 Formes équivalentes de la relation «inclusion»

La relation d'ordre partiel peut être exprimée mathématiquement sous des formes différentes mais toutes équivalentes entre elles. Pour cette relation, les huit formulations suivantes (de [2.34] à [2.41]) sont équivalentes pour tout f et g de $\mathbb{I}$.

$$f \le g \qquad [2.34]$$

$$f \cdot g = f \qquad [2.35]$$

$$\overline{f} + g = 1^* \qquad [2.36]$$

$$f \cdot \overline{g} = 0^* \qquad [2.37]$$

$$\overline{g} \cdot \overline{f} = \overline{g} \qquad [2.38]$$

$$f + g = g \qquad [2.39]$$

$$\overline{f} + \overline{g} = \overline{f} \qquad [2.40]$$

$$\overline{g} \le \overline{f} \qquad [2.41]$$

Démonstrations : [2.34] ⇔ [2.35] ⇔ [2.36] ⇔ [2.37] ⇔ [2.38] ⇔ [2.39] ⇔ [2.40] ⇔ [2.41].

Démonstration de l'équivalence [2.34] ⇔ [2.35] : $f \le g \Leftrightarrow f \cdot g = f$

Les 2 relations sont équivalentes par définition, $f \le g \equiv f \cdot g = g$ □

Démonstration de l'équivalence [2.35] ⇔ [2.36] : $f \cdot g = f \Leftrightarrow \overline{f} + g = 1^*$

Cette équivalence sera démontrée en utilisant [2.19],[2.17], [2.8], [2.13] et [2.5], [2.3], [2.7], [2.6] respectivement :

Si $f \cdot g = f$ alors, $\overline{f} + g = \overline{(f \cdot g)} + g = (\overline{f} + \overline{g}) + g = \overline{f} + 1^* = 1^*$, et

Si $\overline{f} + g = 1^*$ alors, $f = f \cdot 1^* = f \cdot (\overline{f} + g) = f \cdot \overline{f} + f \cdot g = 0^* + f \cdot g = f \cdot g$,

donc, $f \cdot g = f \Leftrightarrow \overline{f} + g = 1^*$ □

Démonstration de l'équivalence [2.35] ⇔ [2.37] : $f \cdot g = f \Leftrightarrow f \cdot \overline{g} = 0*$

Cette équivalence sera démontrée en utilisant [2.16], [2.7], [2.12] et [2.5], [2.8], [2.3], [2.6] respectivement :

Si $f \cdot g = f$ alors, $f \cdot \overline{g} = \left(f \cdot g\right) \cdot \overline{g} = 0*$, et

Si $f \cdot \overline{g} = 0*$ alors, $f = f \cdot 1* = f \cdot (g + \overline{g}) = f \cdot g + f \cdot \overline{g} = f \cdot g + 0* = f \cdot g$,

donc, $f \cdot g = f \Leftrightarrow f \cdot \overline{g} = 0*$ □

Démonstration de l'équivalence [2.35] ⇔ [2.38] : $f \cdot g = f \Leftrightarrow \overline{g} \cdot \overline{f} = \overline{g}$

Cette équivalence sera démontrée en utilisant [2.19], [2.15] et [2.18], [2.19], [2.18], [2.15] respectivement :

Si $f \cdot g = f$ alors, $\overline{g} \cdot \overline{f} = \overline{g} \cdot \overline{\left(f \cdot g\right)} = \overline{g} \cdot \left(\overline{f} + \overline{g}\right) = \overline{g}$,

Si $\overline{g} \cdot \overline{f} = \overline{g}$ alors, $f \cdot g = f \cdot \overline{\left(\overline{g}\right)} = f \cdot \overline{\left(\overline{f} \cdot \overline{g}\right)} = f \cdot \left(f + g\right) = f$,

donc, $f \cdot g = f \Leftrightarrow \overline{g} \cdot \overline{f} = \overline{g}$ □

Démonstration de l'équivalence [2.35] ⇔ [2.39] : $f \cdot g = f \Leftrightarrow f + g = g$

Cette équivalence sera démontrée en utilisant [2.14] et [2.15] respectivement :

Si $f \cdot g = f$ alors, $f + g = (f \cdot g) + g = g$,

Si $f + g = g$ alors, $f \cdot g = f \cdot (f + g) = f$,

donc, $f \cdot g = f \Leftrightarrow f + g = g$ □

Démonstration de l'équivalence [2.38] ⇔ [2.40] : $\overline{g} \cdot \overline{f} = \overline{g} \Leftrightarrow \overline{f} + \overline{g} = \overline{f}$

Cette équivalence sera démontrée en utilisant [2.14] et [2.15] respectivement :

Si $\overline{g} \cdot \overline{f} = \overline{g}$ alors, $\overline{f} + \overline{g} = \overline{f} + (\overline{g} \cdot \overline{f}) = \overline{f}$,

Si $\overline{f} + \overline{g} = \overline{f}$ alors, $\overline{g} \cdot \overline{f} = \overline{g} \cdot (\overline{f} + \overline{g}) = \overline{g}$,

donc, $\overline{g} \cdot \overline{f} = \overline{g} \Leftrightarrow \overline{f} + \overline{g} = \overline{f}$ □

Démonstration de l'équivalence [2.35] ⇔ [2.41] : $f \cdot g = f \Leftrightarrow \overline{g} \leq \overline{f}$

Cette équivalence sera démontrée en utilisant l'équivalence [2.35] ⇔ [2.38] :

Comme $\overline{g} \leq \overline{f} \Leftrightarrow \overline{g} \cdot \overline{f} = \overline{g}$ par définition, et $\overline{g} \cdot \overline{f} = \overline{g} \Leftrightarrow f \cdot g = f$

alors, par transitivité $\overline{g} \leq \overline{f} \Leftrightarrow f \cdot g = f$ □

Par transitivité de la relation d'équivalence, les formulations [2.34] à [2.41] sont bien toutes équivalentes entre elles. Elles représentent la même relation d'ordre partiel entre les signaux f et g.

2.3.2.3 Théorèmes relatifs à la relation «inclusion»

Il existe d'autres théorèmes qui sont respectés pour toute relation d'ordre partiel. D'abord nous allons donner des théorèmes d'implication : [2.42] à [2.45]. Ensuite nous allons donner des théorèmes d'équivalence : [2.46] à [2.53].

Les implications suivantes sont vérifiées pour tout signal f, g h, et i de $\mathbb{I}$.

$$f \le g \quad \Rightarrow \quad (f \cdot h) \le (g \cdot h) \qquad [2.42]$$

$$f \le g \quad \Rightarrow \quad (f + h) \le (g + h) \qquad [2.43]$$

$$\begin{cases} f \le g \\ h \le i \end{cases} \Rightarrow \quad (f \cdot h) \le (g \cdot i) \qquad [2.44]$$

$$\begin{cases} f \le g \\ h \le i \end{cases} \Rightarrow \quad (f + h) \le (g + i) \qquad [2.45]$$

Pour la relation "Inclusion" les équivalences suivantes sont également vérifiées.

$$\begin{cases} f \le h \\ g \le h \end{cases} \quad \Leftrightarrow \quad (f + g) \le h \qquad [2.46]$$

$$\begin{cases} f \le g \\ f \le h \end{cases} \quad \Leftrightarrow \quad f \le (g \cdot h) \qquad [2.47]$$

$$(f \cdot g) \le h \quad \Leftrightarrow \quad f \le \left(\overline{g} + h\right) \qquad [2.48]$$

$$\begin{cases} f \le h \\ g \le i \end{cases} \quad \Leftrightarrow \quad \begin{cases} (f \cdot \overline{g}) \le h \\ (\overline{f} \cdot g) \le i \\ (f \cdot g) \le (h \cdot i) \end{cases} \qquad [2.49]$$

$$\begin{cases} (f \cdot \overline{g}) \le h \\ g = 0* \end{cases} \quad \Leftrightarrow \quad \begin{cases} f \le h \\ g = 0* \end{cases} \qquad [2.50]$$

Les trois cas suivants sont équivalents :

$$(f \cdot \overline{h}) \le h \quad \Leftrightarrow \quad f \le h \qquad [2.51]$$

$$(f + g \cdot h) \le h \quad \Leftrightarrow \quad f \le h \qquad [2.52]$$

$$f \le (f \cdot h) \quad \Leftrightarrow \quad f \le h \qquad [2.53]$$

2.3.2.4 Démonstrations des implications et équivalences [2.42] à [2.53]

> **Note** : Cette sous-section peut être omise en première lecture

Démonstration de l'implication [2.42] : $f \le g \quad \Rightarrow \quad (f \cdot h) \le (g \cdot h)$

Si $f \le g$, nous avons $f \cdot g = f$; de sorte que (en utilisant [2.16], [2.1], [2.16], [2.10])

$$(f \cdot h) \cdot (g \cdot h) = f \cdot h \cdot g \cdot h = (f \cdot g) \cdot (h \cdot h) = (f \cdot g) \cdot (h)$$
$$= f \cdot h$$

donc, $(f \cdot h) \le (g \cdot h)$

alors, $f \le g \Rightarrow (f \cdot h) \le (g \cdot h)$ □

Démonstration de l'implication [2.43] : $f \le g \quad \Rightarrow \quad (f + h) \le (g + h)$

Si $f \le g$, nous avons $f \cdot g = f$; de sorte que (en utilisant [2.4])

$$(f + h) \cdot (g + h) = (f \cdot g) + h$$
$$= f + h$$

donc, $(f + h) \le (g + h)$

alors, $f \le g \Rightarrow (f + h) \le (g + h)$ □

Démonstration de l'implication [2.44] : $\begin{cases} f \le g \\ h \le i \end{cases} \Rightarrow \quad (f \cdot h) \le (g \cdot i)$

Si $f \le g$ et $h \le i$, nous avons $f \cdot g = f$ et $h \cdot i = h$; de sorte que (en utilisant [2.16], [2.1], [2.16])

$$(f \cdot h) \cdot (g \cdot i) = (f \cdot h \cdot g \cdot i) = (f \cdot g) \cdot (h \cdot i) = (f) \cdot (h)$$
$$= f \cdot h$$

donc, $(f \cdot h) \le (g \cdot i)$

alors, $\begin{cases} f \le g \\ h \le i \end{cases} \Rightarrow (f \cdot h) \le (g \cdot i)$ □

Démonstration de l'implication [2.45] : $\begin{cases} f \le g \\ h \le i \end{cases} \Rightarrow \quad (f + h) \le (g + i)$

Si $f \le g$ et $h \le i$, nous avons $f \cdot g = f$ et $h \cdot i = h$; de sorte que (en utilisant [2.3], [2.14])

$$(f + h) \cdot (g + i) = (f \cdot g) + (f \cdot i) + (g \cdot h) + (h \cdot i)$$
$$= (f) + (f \cdot i) + (g \cdot h) + (h)$$
$$= f + h$$

donc, $(f+h) \le (g+i)$

alors, $\begin{cases} f \le g \\ h \le i \end{cases} \Rightarrow (f+h) \le (g+i)$ □

Démonstration de l'équivalence [2.46] : $\begin{cases} f \le h \\ g \le h \end{cases} \Leftrightarrow (f+g) \le h$

Cette équivalence sera démontrée en utilisant [2.34] ⇔ [2.37], [2.25], [2.3], [2.37] ⇔ [2.34]:

$$\begin{cases} f \le h \\ g \le h \end{cases} \Leftrightarrow \begin{cases} f \cdot \overline{h} = 0 * \\ g \cdot \overline{h} = 0 * \end{cases}$$
$$\Leftrightarrow f \cdot \overline{h} + g \cdot \overline{h} = 0 *$$
$$\Leftrightarrow (f+g) \cdot \overline{h} = 0 *$$
$$\Leftrightarrow (f+g) \le h$$ □

Démonstration de l'équivalence [2.47] : $\begin{cases} f \le g \\ f \le h \end{cases} \Leftrightarrow f \le (g \cdot h)$

Cette équivalence sera démontrée en utilisant [2.34] ⇔ [2.37], [2.25], [2.3], [2.37] ⇔ [2.34], [2.20], [2.18] :

$$\begin{cases} f \le g \\ f \le h \end{cases} \Leftrightarrow \begin{cases} f \cdot \overline{g} = 0 * \\ f \cdot \overline{h} = 0 * \end{cases}$$
$$\Leftrightarrow f \cdot \overline{g} + f \cdot \overline{h} = 0 *$$
$$\Leftrightarrow f \cdot (\overline{g} + \overline{h}) = 0 *$$
$$\Leftrightarrow f \le (g \cdot h)$$ □

Démonstration de l'équivalence [2.48] : $(f \cdot g) \le h \Leftrightarrow f \le \left(\overline{g} + h\right)$

Cette équivalence sera démontrée en utilisant [2.34] ⇔ [2.37], [2.16], [2.37] ⇔ [2.34], [2.19], [2.18] :

$$(f \cdot g) \le h \Leftrightarrow (f \cdot g) \cdot \overline{h} = 0 *$$
$$\Leftrightarrow f \cdot (g \cdot \overline{h}) = 0 *$$
$$\Leftrightarrow f \le \left(\overline{g} + h\right)$$ □

Démonstration de l'équivalence [2.49] : $\begin{cases} f \le h \\ g \le i \end{cases} \Leftrightarrow \begin{cases} (f \cdot \overline{g}) \le h \\ (\overline{f} \cdot g) \le i \\ (f \cdot g) \le (h \cdot i) \end{cases}$

Cette équivalence sera démontrée en utilisant [2.34] ⇔ [2.37], [2.25], [2.5], [2.8], [2.3], [2.3], [2.25], [2.37] ⇔ [2.34], [2.20], [2.18] :

$$\begin{cases} f \le h \\ g \le i \end{cases} \Leftrightarrow \begin{cases} f \cdot \overline{h} = 0* \\ g \cdot \overline{i} = 0* \end{cases}$$
$$\Leftrightarrow \quad f \cdot \overline{h} + g \cdot \overline{i} = 0*$$
$$\Leftrightarrow \quad f \cdot \overline{h} \cdot 1* + g \cdot \overline{i} \cdot 1* = 0*$$
$$\Leftrightarrow \quad f \cdot \overline{h} \cdot (g + \overline{g}) + g \cdot \overline{i} \cdot (f + \overline{f}) = 0*$$
$$\Leftrightarrow \quad f \cdot g \cdot \overline{h} + f \cdot \overline{g} \cdot \overline{h} + f \cdot g \cdot \overline{i} + \overline{f} \cdot g \cdot \overline{i} = 0*$$
$$\Leftrightarrow \quad f \cdot \overline{g} \cdot \overline{h} + \overline{f} \cdot g \cdot \overline{i} + f \cdot g \cdot (\overline{h} + \overline{i}) = 0*$$
$$\Leftrightarrow \begin{cases} f \cdot \overline{g} \cdot \overline{h} = 0* \\ \overline{f} \cdot g \cdot \overline{i} = 0* \\ f \cdot g \cdot (\overline{h} + \overline{i}) = 0* \end{cases}$$
$$\Leftrightarrow \begin{cases} (f \cdot \overline{g}) \le h \\ (\overline{f} \cdot g) \le i \\ (f \cdot g) \le (h \cdot i) \end{cases}$$

□

Démonstration de l'équivalence [2.50] : $\begin{cases} (f \cdot \overline{g}) \le h \\ g = 0* \end{cases} \Leftrightarrow \begin{cases} f \le h \\ g = 0* \end{cases}$

Cette équivalence sera démontrée en utilisant [2.34] $\Leftrightarrow$ [2.37] ; [2.25] ; [2.21] ; [2.25] et [2.37] $\Leftrightarrow$ [2.34] :

$$\begin{cases} f \cdot \overline{g} \le h \\ g = 0* \end{cases} \Leftrightarrow \begin{cases} f \cdot \overline{g} \cdot \overline{h} = 0* \\ g = 0* \end{cases} \Leftrightarrow \begin{cases} f \cdot \overline{g} \cdot \overline{h} + g = 0* \end{cases}$$
$$\Leftrightarrow \begin{cases} f \cdot \overline{h} + g = 0* \end{cases} \Leftrightarrow \begin{cases} f \cdot \overline{h} = 0* \\ g = 0* \end{cases}$$
$$\Leftrightarrow \begin{cases} f \le h \\ g = 0* \end{cases}$$

□

Démonstration de l'équivalence [2.51] : $(f \cdot \overline{h}) \le h \quad \Leftrightarrow \quad f \le h$

Cette équivalence sera démontrée en utilisant [2.34] $\Leftrightarrow$[2.37], [2.16], [2.10], [2.37] $\Leftrightarrow$[2.34]:

$$(f \cdot \overline{h}) \le h \quad \Leftrightarrow \quad (f \cdot \overline{h}) \cdot \overline{h} = 0*$$
$$\Leftrightarrow \quad f \cdot \overline{h} = 0*$$
$$\Leftrightarrow \quad f \le h$$

□

Démonstration de l'équivalence [2.52] : $(f + g \cdot h) \le h \quad \Leftrightarrow \quad f \le h$

Cette équivalence sera démontrée en utilisant [2.46] :

$$(f + g \cdot h) \le h \quad \Leftrightarrow \quad \begin{cases} f \le h \\ g \cdot h \le h \end{cases}$$

comme $g \cdot h \leq h$ est toujours vraie alors

$\Leftrightarrow \quad f \leq h$ □

Démonstration de l'équivalence [2.53] : $f \leq (f \cdot h) \quad \Leftrightarrow \quad f \leq h$

Cette équivalence sera démontrée en utilisant [2.34] ⇔[2.35], [2.16], [2.10], [2.34] ⇔[2.35]:

$$\begin{aligned} f \leq (f \cdot h) \quad &\Leftrightarrow \quad f \cdot (f \cdot h) = f \\ &\Leftrightarrow \quad (f \cdot h) = f \\ &\Leftrightarrow \quad f \leq h \end{aligned}$$ □

2.3.3 Relation d'égalité

Toute relation "Egalité" peut être également exprimée à l'aide de la relation d'ordre partiel car les deux formulations suivantes ([2.54] et [2.55]) sont équivalentes.

$$f = g \qquad [2.54]$$

$$\begin{cases} f \leq g \\ \overline{f} \leq \overline{g} \end{cases} \qquad [2.55]$$

D'ailleurs, pour la relation "Egalité", les sept formulations suivantes (de [2.56] à [2.62]) sont aussi équivalentes à [2.54] et [2.55], pour tout f, g et $h \in \mathbb{I}$.

$$\overline{f} = \overline{g} \qquad [2.56]$$

$$\begin{cases} f \cdot \overline{g} = 0 * \\ \overline{f} \cdot g = 0 * \end{cases} \qquad [2.57]$$

$$f \cdot \overline{g} + \overline{f} \cdot g = 0 * \qquad [2.58]$$

$$f \cdot g + \overline{f} \cdot \overline{g} = 1 * \qquad [2.59]$$

$$\begin{cases} f + \overline{g} = 1 * \\ f \cdot \overline{g} = 0 * \end{cases} \qquad [2.60]$$

$$\begin{cases} f \cdot h = g \cdot h \\ f \cdot \overline{h} = g \cdot \overline{h} \end{cases} \qquad [2.61]$$

$$\begin{cases} f \cdot h = g \cdot h \\ f + h = g + h \end{cases} \qquad [2.62]$$

Nous présentons ci-dessous un cas particulier de l'égalité :

$$f = 0^* \quad \Leftrightarrow \quad f \le 0^* \qquad [2.63]$$

Les démonstrations des équivalences entre ces formulations ([2.54] à [2.62]) sont données ci-après. La démonstration de [2.63] est donnée ensuite.

Démonstration de l'équivalence [2.54] $\Leftrightarrow$ [2.55] : $f = g \Leftrightarrow \begin{cases} f \le g \\ \overline{f} \le \overline{g} \end{cases}$

A partir de la propriété de l'anti-symétrie [2.29] et l'équivalence [2.34] $\Leftrightarrow$ [2.41]

Si $\begin{cases} f \le g \\ g \le f \end{cases} \Rightarrow f = g$, alors $\begin{cases} f \le g \\ \overline{f} \le \overline{g} \end{cases} \Rightarrow f = g$.

A partir de la propriété de la réflexivité [2.27] de f et g, et en utilisant [2.34] $\Leftrightarrow$ [2.41]

Comme [2.27] est toujours vrai, alors $\begin{cases} f \le f \\ g \le g \end{cases}$ l'est aussi ; de sorte que si $f = g$, nous avons $\begin{cases} f \le (f) \\ g \le (g) \end{cases} \Leftrightarrow \begin{cases} f \le g \\ g \le f \end{cases} \Leftrightarrow \begin{cases} f \le g \\ \overline{f} \le \overline{g} \end{cases}$ donc, $f = g \Leftrightarrow \begin{cases} f \le g \\ \overline{f} \le \overline{g} \end{cases}$ □

Démonstration de l'équivalence [2.55] $\Leftrightarrow$ [2.56] : $\begin{cases} f \le g \\ \overline{f} \le \overline{g} \end{cases} \Leftrightarrow \overline{f} = \overline{g}$

Cette équivalence sera démontrée en utilisant [2.54] $\Leftrightarrow$ [2.55], [2.18] et [2.1]

$$\overline{f} = \overline{g} \quad \Leftrightarrow \quad \begin{cases} (\overline{f}) \le (\overline{g}) \\ \overline{(\overline{f})} \le \overline{(\overline{g})} \end{cases} \quad \Leftrightarrow \quad \begin{cases} \overline{f} \le \overline{g} \\ f \le g \end{cases} \quad \Leftrightarrow \quad \begin{cases} f \le g \\ \overline{f} \le \overline{g} \end{cases}$$ □

Démonstration de l'équivalence [2.55] $\Leftrightarrow$ [2.57] : $\begin{cases} f \le g \\ \overline{f} \le \overline{g} \end{cases} \Leftrightarrow \begin{cases} f \cdot \overline{g} = 0^* \\ \overline{f} \cdot g = 0^* \end{cases}$

Cette équivalence sera démontrée en utilisant [2.34] $\Leftrightarrow$ [2.37], [2.18]

$$\begin{cases} f \le g \\ \overline{f} \le \overline{g} \end{cases} \quad \Leftrightarrow \quad \begin{cases} f \cdot \overline{g} = 0^* \\ \overline{f} \cdot g = 0^* \end{cases}$$ □

Démonstration de l'équivalence [2.57] $\Leftrightarrow$ [2.58] : $\begin{cases} f \cdot \overline{g} = 0^* \\ \overline{f} \cdot g = 0^* \end{cases} \Leftrightarrow f \cdot \overline{g} + \overline{f} \cdot g = 0^*$

Cette équivalence sera démontrée en utilisant [2.25]

$$\begin{cases} f \cdot \overline{g} = 0^* \\ \overline{f} \cdot g = 0^* \end{cases} \Leftrightarrow \quad f \cdot \overline{g} + \overline{f} \cdot g = 0^* \qquad \square$$

Démonstration de l'équivalence [2.55] $\Leftrightarrow$ [2.59] : $\begin{cases} f \le g \\ \overline{f} \le \overline{g} \end{cases} \Leftrightarrow f \cdot g + \overline{f} \cdot \overline{g} = 1^*$

Cette équivalence sera démontrée en utilisant [2.34] $\Leftrightarrow$ [2.36], [2.18] et [2.26], [2.3], [2.7] et [2.6] puis [2.2]

$$\begin{cases} f \le g \\ \overline{f} \le \overline{g} \end{cases} \Leftrightarrow \begin{cases} (\overline{f} + g) = 1^* \\ (\overline{\overline{f}} + \overline{g}) = 1^* \end{cases}$$
$$\Leftrightarrow (\overline{f} + g) \cdot (f + \overline{g}) = 1^*$$
$$\Leftrightarrow (\overline{f} \cdot f) + (\overline{f} \cdot \overline{g}) + (f \cdot g) + (g \cdot \overline{g}) = 1^*$$
$$\Leftrightarrow (\overline{f} \cdot \overline{g}) + (f \cdot g) = 1^*$$
$$\Leftrightarrow f \cdot g + \overline{f} \cdot \overline{g} = 1^* \qquad \square$$

Démonstration de l'équivalence [2.55] $\Leftrightarrow$ [2.60] : $\begin{cases} f \le g \\ \overline{f} \le \overline{g} \end{cases} \Leftrightarrow \begin{cases} f + \overline{g} = 1^* \\ f \cdot \overline{g} = 0^* \end{cases}$

Cette équivalence sera démontrée en utilisant [2.34] $\Leftrightarrow$ [2.37], [2.34] $\Leftrightarrow$ [2.36] et [2.18]

$$\begin{cases} f \le g \\ \overline{f} \le \overline{g} \end{cases} \Leftrightarrow \begin{cases} (f \cdot \overline{g}) = 0^* \\ (\overline{\overline{f}} + \overline{g}) = 1^* \end{cases} \Leftrightarrow \begin{cases} (f \cdot \overline{g}) = 0^* \\ (f + \overline{g}) = 1^* \end{cases} \qquad \square$$

Démonstration de l'équivalence [2.54] $\Leftrightarrow$ [2.61] : $f = g \Leftrightarrow \begin{cases} f \cdot h = g \cdot h \\ f \cdot \overline{h} = g \cdot \overline{h} \end{cases}$

Cette équivalence sera démontrée en utilisant [2.54] $\Leftrightarrow$ [2.59]; [2.16], [2.1], [2.16], [2.10], [2.19], [2.4], [2.18]; [2.17], [2.21], [2.2], [2.17]; [2.26]; [2.4], [2.7], [2.6]; [2.59] $\Leftrightarrow$ [2.54]

$$\begin{cases} f \cdot h = g \cdot h \\ f \cdot \overline{h} = g \cdot \overline{h} \end{cases} \Leftrightarrow \begin{cases} (f \cdot h) \cdot (g \cdot h) + (\overline{f \cdot h}) \cdot (\overline{g \cdot h}) = 1^* \\ (f \cdot \overline{h}) \cdot (g \cdot \overline{h}) + \left(\overline{f \cdot \overline{h}}\right) \cdot \left(\overline{g \cdot \overline{h}}\right) = 1^* \end{cases}$$
$$\Leftrightarrow \begin{cases} f \cdot g \cdot h + (\overline{h} + \overline{f} \cdot \overline{g}) = 1^* \\ f \cdot g \cdot \overline{h} + (h + \overline{f} \cdot \overline{g}) = 1^* \end{cases}$$
$$\Leftrightarrow \begin{cases} \overline{h} + (f \cdot g + \overline{f} \cdot \overline{g}) = 1^* \\ h + (f \cdot g + \overline{f} \cdot \overline{g}) = 1^* \end{cases}$$
$$\Leftrightarrow \left\{ \left(\overline{h} + (f \cdot g + \overline{f} \cdot \overline{g})\right) \cdot \left(h + (f \cdot g + \overline{f} \cdot \overline{g})\right) = 1^* \right.$$
$$\Leftrightarrow \left\{ f \cdot g + \overline{f} \cdot \overline{g} = 1^* \right.$$
$$\Leftrightarrow f = g \qquad \square$$

Démonstration de l'équivalence [2.54] ⇔ [2.62] : $f = g \Leftrightarrow \begin{cases} f \cdot h = g \cdot h \\ f + h = g + h \end{cases}$

Cette équivalence sera démontrée en utilisant [2.54] ⇔ [2.58]; [2.19], [2.20], [2.1]; [2.3], [2.7], [2.6]; [2.25], [2.3]; [2.8], [2.5]; [2.58] ⇔ [2.54]

$$\begin{cases} f \cdot h = g \cdot h \\ f + h = g + h \end{cases} \Leftrightarrow \begin{cases} (f \cdot h) \cdot (\overline{g \cdot h}) + (\overline{f \cdot h}) \cdot (g \cdot h) = 0* \\ (f + h) \cdot (\overline{g + h}) + \overline{(f + h)} \cdot (g + h) = 0* \end{cases}$$

$$\Leftrightarrow \begin{cases} f \cdot h \cdot (\overline{g} + \overline{h}) + (\overline{f} + \overline{h}) \cdot g \cdot h = 0* \\ (f + h) \cdot (\overline{g} \cdot \overline{h}) + (\overline{f} \cdot \overline{h}) \cdot (g + h) = 0* \end{cases}$$

$$\Leftrightarrow \begin{cases} f \cdot \overline{g} \cdot h + \overline{f} \cdot g \cdot h = 0* \\ f \cdot \overline{g} \cdot \overline{h} + \overline{f} \cdot g \cdot \overline{h} = 0* \end{cases}$$

$$\Leftrightarrow \left\{ f \cdot \overline{g} \cdot (h + \overline{h}) + \overline{f} \cdot g \cdot (h + \overline{h}) = 0* \right.$$

$$\Leftrightarrow \left\{ f \cdot \overline{g} + \overline{f} \cdot g = 0* \right.$$

$$\Leftrightarrow f = g$$

□

Par transitivité de la relation d'équivalence, les formulations [2.54] à [2.62] sont bien toutes équivalentes entre elles. Elles représentent bien la même relation entre les signaux f et g.

Démonstration de l'équivalence [2.63] : $f = 0* \quad \Leftrightarrow \quad f \leq 0*$

Cette équivalence sera démontrée en utilisant [2.54] ⇔ [2.55], [2.24] :

$$f = 0* \quad \Leftrightarrow \quad \begin{cases} f \leq 0* \\ \overline{f} \leq \overline{0*} \end{cases} \quad \Leftrightarrow \quad \begin{cases} f \leq 0* \\ \overline{f} \leq 1* \end{cases}$$

comme $\overline{f} \leq 1*$ est toujours vraie ([2.30]) alors

$$\Leftrightarrow \quad f \leq 0*$$

□

2.4 Conclusion

Nous venons d'établir les bases de l'algèbre $\mathbb{I}$, cadre formel de nos travaux, et de démontrer des théorèmes qui serviront pour la conception de Systèmes à Evénements Discrets. Ces théorèmes portent sur les opérations de base ET, OU, NON et sur les relations "=" et "≤".
Dans le chapitre suivant, nous introduisons de nouvelles opérations permettant de décrire des comportements séquentiels et temporisés.

Chapitre 3

Nous présentons dans ce chapitre les opérations de l'algèbre 𝕀 permettant de décrire des comportements séquentiels et temporisés, à savoir :

- Les opérations Fronts : RE(↑) et FE(↓)
- Les opérations Mémoires : SR et RS
- Les opérations Temporisateurs : TON et TOF

Les théorèmes relatifs à ces opérations sont également présentés.

3. Opérations de l'algèbre 𝕀 permettant de décrire des événements et comportements séquentiels et temporisés

3.1 Introduction

Rappelons que le cadre mathématique utilisé dans ce document (algèbre 𝕀) est conçu pour l'étude des systèmes logiques, c'est-à-dire les systèmes pour lesquels les variables ne peuvent prendre que les deux seules valeurs 0 ou 1. Les opérations permettant de décrire des comportements combinatoires ont été décrites au chapitre précédent.

Dans ce chapitre, nous introduisons des opérations permettant de décrire des événements (↑ et ↓ à la section 3.2), des comportements séquentiels (mémoires SR et RS à la section 3.3) et temporisés (temporisations TON et TOF à la section 3.4).

3.2 Opérations permettant de décrire des événements

3.2.1 Introduction

L'algèbre 𝕀 a été définie initialement dans [Rou 93] afin de combler l'absence de définition formelle à la notion d'événements telle qu'elle est pratiquée en Grafcet. En effet, cette notion s'avère insuffisante pour être exempte d'ambiguïté pour évaluer des équations logiques car les réceptivités comprenant des aspects temporels tels que les événements ne sont pas définis au sens de l'algèbre de Boole [Bou 92].

D'un point de vue théorique, la norme [IEC 61131-3], relative aux langages de programmation, présente les événements sous forme d'équations récurrentes qui reposent sur deux valeurs consécutives d'une variable (voir annexe A). Cette définition à l'aide d'une formule de récurrence, fonction d'un moniteur échantillonnant un signal, n'est pas nécessaire. Ainsi, deux opérations : l'opération front montant et l'opération front descendant, ont été définies formellement sur l'algèbre 𝕀 pour la représentation formelle des événements.

3.2.2 Définition mathématique des opérations Fronts

Pour décrire la notion d'événements, deux opérations spécifiques ont été définies au sein de l'algèbre $\mathbb{I}$. La définition mathématique de ces opérations est la suivante :

Définition 7 : Opération RE

$$\begin{aligned} \mathbb{I} &\rightarrow \mathbb{I} \\ f &\mapsto \uparrow f \end{aligned}$$

Définition 8 : Opération FE

$$\begin{aligned} \mathbb{I} &\rightarrow \mathbb{I} \\ f &\mapsto \downarrow f \end{aligned}$$

Où $\forall t \in \mathbb{R}^{+*}$,

$$\uparrow f(t) = f(t) \wedge \left[\exists \varepsilon_0 > 0 : \forall \varepsilon \in]0, \varepsilon_0[: f(t-\varepsilon) = 0 \right]$$

$$\downarrow f(t) \ = \neg f(t) \wedge \left[\exists \varepsilon_0 > 0 : \forall \varepsilon \in]0, \varepsilon_0[: f(t-\varepsilon) = 1 \right]$$

Il est représenté sur la figure 17, un signal f de $\mathbb{I}$ et le résultat du traitement de ce signal par les opérations RE et FE.

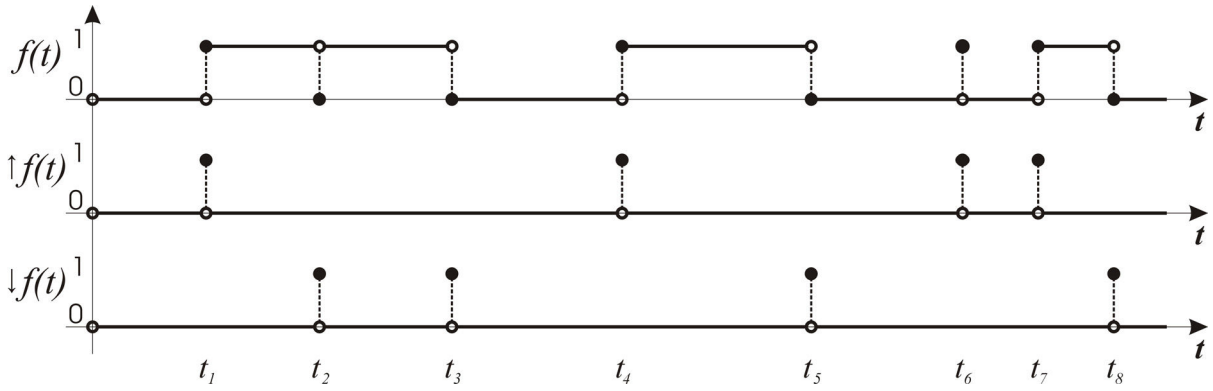


Figure 17 *Représentation graphique des fonctions fronts*

L'opération RE (*Rising Edge*) permet de construire un signal binaire $\uparrow f$ à partir de tout signal f. Le signal $\uparrow f$ est vrai pour les seules dates où le signal f passe de la valeur 0 à la valeur 1. Le signal $\uparrow f$ est construit à partir des informations présentes dans le signal f de la façon suivante. Pour toute date t de $\mathbb{R}^{+*}$, la valeur $\uparrow f(t)$ de la fonction $\uparrow f$ est obtenue par le ET logique entre deux variables logiques. La première variable logique est la valeur de la fonction f à cette même date t tandis que la seconde variable logique est le résultat d'un prédicat. Ce prédicat est vrai s'il existe un intervalle ouvert de longueur ε_0 précédent la date t sur lequel la valeur du signal f vaut toujours 0.

Pour le signal binaire présenté sur la figure 17, nous avons :

- $f(t)=1$ pour $t \in [t_1,t_2[\cup]t_2,t_3[\cup [t_4,t_5[\cup \{t_6\} \cup [t_7,t_8[$
- le prédicat $\left[\exists \varepsilon_0 > 0 : \forall \varepsilon \in]0,\varepsilon_0[: f(t-\varepsilon)=0\right]$ est vrai pour

  $t \in]0,t_1] \cup]t_3,t_4] \cup]t_5,t_7] \cup]t_8,+\infty[$

Le signal $\uparrow f$ est donc vrai aux seules dates $t \in \{t_1\} \cup \{t_4\} \cup \{t_6\} \cup \{t_7\}$.

L'opération FE (*Falling Edge*) permet de construire un signal binaire $\downarrow f$ à partir de tout signal f. Le signal $\downarrow f$ est vrai pour les seules dates où le signal f passe de la valeur 1 à la valeur 0. Le signal $\downarrow f$ est construit à partir des informations présentes dans le signal f de la façon suivante. Pour toute date t de $\mathbb{R}^{+*}$, la valeur $\downarrow f(t)$ de la fonction $\downarrow f$ est obtenue par le ET logique entre deux variables logiques. La première variable logique est le complément de la valeur de la fonction f à cette même date t tandis que la seconde est le résultat d'un prédicat. Ce prédicat est vrai s'il existe un intervalle ouvert de longueur ε_0 précédent la date t sur lequel la valeur du signal f vaut toujours 1.

Pour le signal binaire présenté sur la figure 17, nous avons :

- $f(t)=0$ pour $t \in]0,t_1[\cup \{t_2\} \cup [t_3,t_4[\cup [t_5,t_6[\cup]t_6,t_7[\cup [t_8,+\infty[$
- le prédicat $\left[\exists \varepsilon_0 > 0 : \forall \varepsilon \in]0,\varepsilon_0[: f(t-\varepsilon)=1\right]$ est vrai pour $t \in]t_1,t_3] \cup]t_4,t_5] \cup]t_7,t_8]$

Le signal $\downarrow f$ est donc vrai au seules dates $t \in \{t_2\} \cup \{t_3\} \cup \{t_5\} \cup \{t_8\}$.

3.2.3 Théorèmes relatifs aux opérations RE et FE

A partir de la définition mathématique des opérations fronts, il est possible de démontrer les théorèmes suivants :

$$\uparrow f \leq f \qquad [3.1]$$

$$\downarrow f \leq \overline{f} \qquad [3.2]$$

$$\uparrow \overline{f} = \downarrow f \qquad [3.3]$$

$$\downarrow \overline{f} = \uparrow f \qquad [3.4]$$

$$\uparrow \left(\uparrow f\right) = \uparrow f \qquad [3.5]$$

$$\uparrow \left(\downarrow f\right) = \downarrow f \qquad [3.6]$$

$$\downarrow\left(\uparrow f\right) = 0* \qquad [3.7]$$

$$\downarrow\left(\downarrow f\right) = 0* \qquad [3.8]$$

$$\uparrow\left(\prod_{i\in\{1,n\}} f_i\right) = \sum_{i\in\{1,n\}}\left(\uparrow f_i \cdot \prod_{\substack{j\in\{1,n\}\\ j\neq i}} f_j\right) \qquad [3.9]$$

$$\uparrow\left(\sum_{i\in\{1,n\}} f_i\right) = \sum_{i\in\{1,n\}}\left(\uparrow f_i \cdot \prod_{\substack{j\in\{1,n\}\\ j\neq i}} \left(\uparrow f_j + (\overline{f_j}\cdot\overline{\downarrow f_j})\right)\right) \qquad [3.10]$$

$$\downarrow\left(\prod_{i\in\{1,n\}} f_i\right) = \sum_{i\in\{1,n\}}\left(\downarrow f_i \cdot \prod_{\substack{j\in\{1,n\}\\ j\neq i}} \left(\downarrow f_j + (f_j\cdot\overline{\uparrow f_j})\right)\right) \qquad [3.11]$$

$$\downarrow\left(\sum_{i\in\{1,n\}} f_i\right) = \sum_{i\in\{1,n\}}\left(\downarrow f_i \cdot \prod_{\substack{j\in\{1,n\}\\ j\neq i}} \overline{f_j}\right) \qquad [3.12]$$

3.2.4 Démonstrations des théorèmes relatifs aux opérations RE et FE

Seuls les théorèmes [3.1] et [3.2] seront démontrés dans ce mémoire car le lecteur pourra se reporter à [Rou 93] pour la démonstration complète des autres théorèmes ([3.3] à [3.12]).

Démonstration de la relation [3.1] : $\uparrow f \leq f$

Ce théorème sera démontré en utilisant sa forme équivalente ([2.34] $\Leftrightarrow$ [2.37]) : $\uparrow f \cdot \overline{f} = 0*$

$\forall t \in \mathbb{R}^{+*}$,

$$\begin{aligned}(\overline{f}\cdot\uparrow f)(t) &= \overline{f}(t) \wedge (\uparrow f)(t)\\ &= \neg f(t) \wedge \left(f(t) \wedge \left[\exists \varepsilon_0 > 0 : \forall \varepsilon \in]0, \varepsilon_0[: f(t-\varepsilon) = 0\right]\right)\\ &= \neg f(t) \wedge f(t) \wedge \left[\exists \varepsilon_0 > 0 : \forall \varepsilon \in]0, \varepsilon_0[: f(t-\varepsilon) = 0\right]\\ &= 0\end{aligned}$$

En conclusion, nous avons bien $\forall t \in \mathbb{R}^{+*},\ (\overline{f}\cdot\uparrow f)(t) = 0 = 0*(t)$

donc, $\overline{f}\cdot\uparrow f = 0*$ □

Démonstration de la relation [3.2] : $\downarrow f \leq \overline{f}$

Ce théorème sera démontré en utilisant sa forme équivalente ([2.34] $\Leftrightarrow$ [2.37]) : $\downarrow f \cdot f = 0*$

$\forall t \in \mathbb{R}^{+*}$,

$$\begin{aligned}(f \cdot \downarrow f)(t) &= f(t) \wedge (\downarrow f)(t) \\ &= f(t) \wedge \left(\neg f(t) \wedge \left[\exists \varepsilon_0 > 0 : \forall \varepsilon \in]0, \varepsilon_0[: f(t-\varepsilon) = 1 \right] \right) \\ &= f(t) \wedge \neg f(t) \wedge \left[\exists \varepsilon_0 > 0 : \forall \varepsilon \in]0, \varepsilon_0[: f(t-\varepsilon) = 1 \right] \\ &= 0\end{aligned}$$

En conclusion, nous avons bien $\forall t \in \mathbb{R}^{+*}$, $(\downarrow f \cdot f)(t) = 0 = 0*(t)$

donc, $\overline{f} \cdot \uparrow f = 0*$ □

3.3 Opérations permettant de décrire des comportements séquentiels

3.3.1 Introduction

Les opérations ET, OU, NON ne permettent que de décrire des comportements combinatoires puisque la valeur courante des résultats ne peut dépendre que de la valeur courante des opérandes. Pour décrire des comportements séquentiels, de nouvelles opérations sont nécessaires. Les deux opérations présentées dans cette section ont été créées à l'origine pour disposer sur *𝕀* d'une définition formelle des blocs fonctionnels mémoires définies dans la norme [IEC 61131-3] consacré aux langages de programmation des Automates Programmables Industriels (voir annexe A).

3.3.2 Définition mathématique des opérations SR, RS

Pour décrire des comportements séquentiels comme celui des mémoires à mise à un prioritaire ou à mise à zéro prioritaire, deux opérations spécifiques ont été définies au sein de l'algèbre *𝕀*. La définition mathématique de ces opérations est la suivante :

Définition 9 : Opération SR

$$\begin{aligned}\mathbb{I}^2 &\rightarrow \mathbb{I} \\ (s,r) &\mapsto SR(s,r)\end{aligned}$$

Où $\forall t \in \mathbb{R}^{+*}$, $SR(s,r)(t) = s(t) \vee \left[\exists t_1 < t : \left(\left(s(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t] : \left((s + \overline{r})(d) = 1 \right) \right) \right]$

Remarque : Les signaux s et r ne sont pas forcement disjoints.

L'opération SR décrit le comportement d'une mémoire à mise à un prioritaire, mise à un par le signal s et remise à zéro par le signal r. Pour toute date t de $\mathbb{R}^{+*}$, la valeur $SR(s,r)(t)$ de la fonction $SR(s,r)$ est obtenue par le OU logique entre deux variables logiques. La première variable logique est la valeur de la fonction s à cette même date t tandis que la seconde est le résultat d'un prédicat. Ce prédicat est vrai s'il existe une date t_1, précédant la date t considérée, pour laquelle la valeur du signal s à cette date t_1 vaut 1 et si pour toute date d telle que $t_1 < d \leq t$ la valeur du signal $s + \bar{r}$ vaut 1.

Définition 10 : Opération RS

$$\begin{array}{rcl} \mathbb{I}^2 & \rightarrow & \mathbb{I} \\ (s,r) & \mapsto & RS(s,r) \end{array}$$

Où $\forall t \in \mathbb{R}^{+*},\ RS(s,r)(t) = \Big(s(t) \wedge \neg r(t)\Big) \vee \Big[\exists t_1 < t : \Big(\big(s(t_1) = 1\big) \wedge \big(\forall d \in [t_1, t] : r(d) = 0\big)\Big)\Big]$

Remarque : Les signaux s et r ne sont pas forcement disjoints.

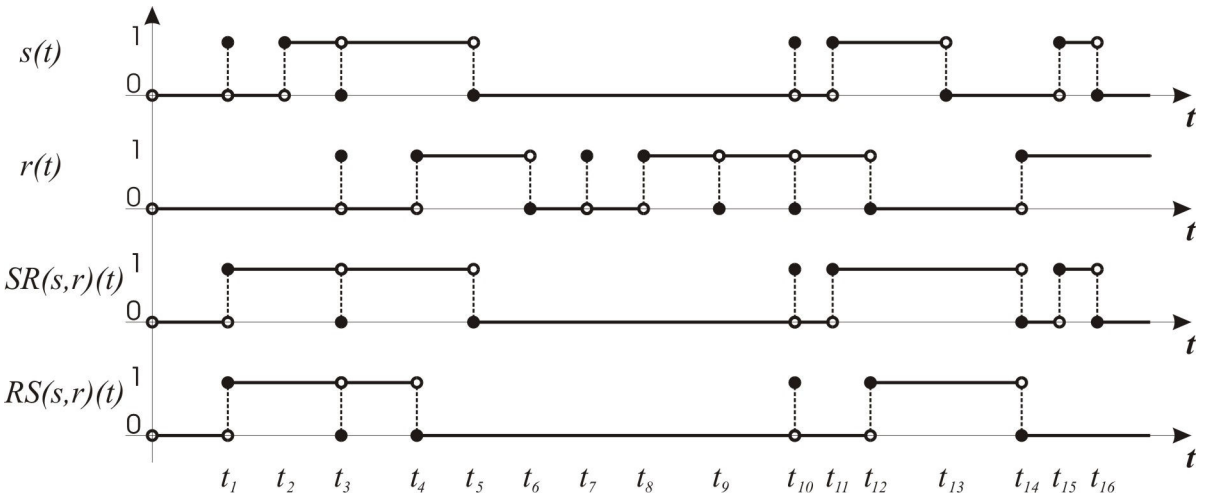


Figure 18 *Représentation graphique des fonctions $SR(s,r)$ et $RS(s,r)$*

L'opération RS décrit le comportement d'une mémoire à mise à zéro prioritaire, mise à un par le signal s et remise à zéro par le signal r. Pour toute date t de $\mathbb{R}^{+*}$, la valeur $RS(s,r)(t)$ de la fonction $RS(s,r)$ est obtenue par le OU logique entre deux variables logiques. La première variable logique est le ET logique entre la valeur de la fonction s à cette même date t et le complément logique de la valeur de la fonction r à cette même date t. La seconde variable logique est le résultat d'un prédicat. Ce prédicat est vrai s'il existe une date t_1 précédant la

date t considérée, pour laquelle la valeur du signal s à cette date t_1 vaut 1 et si pour toute date d telle que $t_1 \le d \le t$ la valeur du signal r vaut 0.

Sur la figure 18, sont représentés deux signaux s et r de $\mathbb{I}$, non disjoints, et le résultat de la composition de ces signaux par les opérations SR et RS.

3.3.3 Théorèmes relatifs aux opérations SR et RS

Il est possible d'établir les théorèmes [3.13] à [3.40], qui font intervenir les opérations SR et RS. Les démonstrations de ces théorèmes sont données à l'annexe D.

$$s \le SR(s,r) \qquad [3.13]$$

$$\bar{s} \cdot r \le \overline{SR(s,r)} \qquad [3.14]$$

$$SR(s,r) = SR(s, \bar{s} \cdot r) \qquad [3.15]$$

$$SR(s,1^*) = s \qquad [3.16]$$

$$SR(s,\bar{s}) = s \qquad [3.17]$$

$$SR(s,s) = SR(s,0^*) \qquad [3.18]$$

$$SR(0^*,r) = 0^* \qquad [3.19]$$

$$SR(1^*,r) = 1^* \qquad [3.20]$$

$$SR(s,r) \cdot r = s \cdot r \qquad [3.21]$$

$$SR(s,r) = SR\left(s,(r + s \cdot f)\right) \qquad [3.22]$$

$$\downarrow SR(s,0^*) = 0^* \qquad [3.23]$$

$$SR(s,r) = RS\left(s,(\bar{s} \cdot r)\right) \qquad [3.24]$$

$$RS(s,r) = SR\left((s \cdot \bar{r}),r\right) \qquad [3.25]$$

$$s \cdot \bar{r} \le RS(s,r) \qquad [3.26]$$

$$r \le \overline{RS(s,r)} \qquad [3.27]$$

$$RS(s,r) = RS\left((s \cdot \bar{r}),r\right) \qquad [3.28]$$

$$RS(s,1^*) = 0^* \qquad [3.29]$$

$$RS(s,\bar{s}) = s \qquad [3.30]$$

$$RS(s,s) = 0* \quad [3.31]$$

$$RS(0*,r) = 0* \quad [3.32]$$

$$RS(1*,r) = \bar{r} \quad [3.33]$$

$$RS(s,r)\cdot s = s\cdot \bar{r} \quad [3.34]$$

$$RS(s,r) = RS\left((s + r\cdot f),r\right) \quad [3.35]$$

$$SR\left(s,(r_1 + r_2)\right) = SR(s,r_1)\cdot SR(s,r_2) \quad [3.36]$$

$$SR\left((s_1 + s_2),r\right) = SR(s_1,r) + SR(s_2,r) \quad [3.37]$$

$$RS\left(s,(r_1 + r_2)\right) = RS(s,r_1)\cdot RS(s,r_2) \quad [3.38]$$

$$RS\left((s_1 + s_2),r\right) = RS(s_1,r) + RS(s_2,r) \quad [3.39]$$

$$RS\left(s_1,(r_1 + s_2)\right)\cdot RS\left(s_2,(r_2 + s_1)\right) = 0* \quad [3.40]$$

3.4 Opérations permettant de décrire des comportements temporisés

3.4.1 Introduction

Pour décrire des comportements temporisés, fonction du temps physique, il est nécessaire d'introduire sur 𝕀 de nouvelles opérations. Les deux opérations présentées dans cette section ont été créées à l'origine pour disposer sur 𝕀 d'une définition formelle des blocs fonctionnels temporisateurs définis dans la norme [IEC 61131-3] consacrée aux langages de programmation des Automates Programmables Industriels (voir annexe A).

Les deux opérations définies dans cette section : le TON et le TOF, permettent d'obtenir des résultats équivalents à ceux présentés dans les chronogrammes de la norme [IEC 61131-3] (voir annexe A). Leur définition mathématique a cependant permis d'établir de nombreux théorèmes qui peuvent être employés dans d'autres modèles.

3.4.2 Définition mathématique des opérations TON et TOF

Pour décrire des comportements temporisés comme celui des temporisateurs à l'enclenchement ou au déclenchement, deux opérations spécifiques ont été définies au sein de l'algèbre $\mathbb{I}$. La définition mathématique de ces opérations est la suivante :

Définition 11 : Opération TON

$$\begin{aligned} \mathbb{I} &\rightarrow \mathbb{I} \\ f &\mapsto d/f \end{aligned}$$

Où $\forall t \in \mathbb{R}^{+*}$, $(d/f)(t) = \begin{cases} \left[0 & \forall t < d \right] \\ \left[\forall t_1 \in]t-d, t] : f(t_1) = 1 & \forall t \geq d \right] \end{cases}$

L'opération TON décrit le comportement d'un temporisateur à l'enclenchement activé par le signal f. Sa définition est donnée par morceaux ($\forall t < d$ et $\forall t \geq d$).

- Pour toute date t de $\mathbb{R}^{+*}$ inférieure à d, la valeur $(d/f)(t)$ de la fonction d/f est 0.
- Pour toute date t de $\mathbb{R}^{+*}$ supérieure à d, la valeur $(d/f)(t)$ de la fonction d/f est le résultat d'un prédicat. Ce prédicat est vrai si pour toute date t_1 telle que $t-d < t_1 \leq t$, la valeur $f(t_1)$ du signal f vaut 1.

La figure 19 montre un signal f de $\mathbb{I}$ et le résultat du traitement de ce signal par l'opération TON.

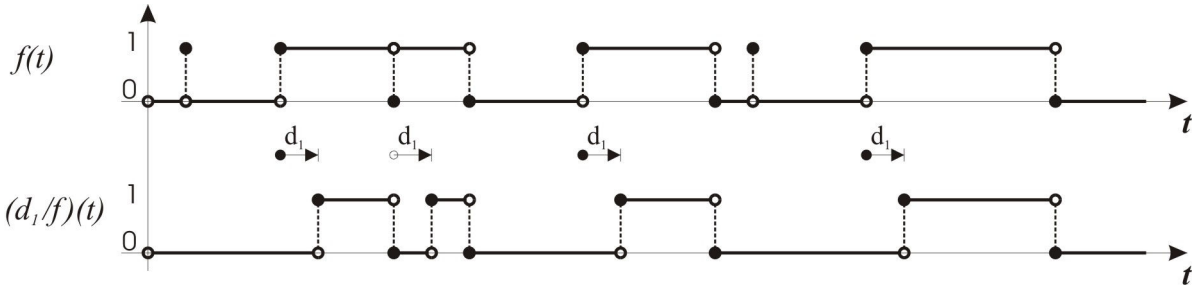


Figure 19 *Représentation graphique de la fonction TON : d_1 / f*

Définition 12 : Opération TOF

$$\begin{aligned} \mathbb{I} &\rightarrow \mathbb{I} \\ f &\mapsto f/d \end{aligned}$$

Où $\forall t \in \mathbb{R}^{+*}$, $(f/d)(t) = \begin{cases} \left[\exists t_1 \in]0,t] : f(t_1) = 1 \qquad \forall t < d\right] \\ \left[\exists t_1 \in]t-d,t] : f(t_1) = 1 \qquad \forall t \geq d\right] \end{cases}$

L'opération TOF décrit le comportement d'un temporisateur au déclenchement activé par le signal f. Comme pour l'opération TON, sa définition est donnée par morceaux ($\forall t < d$ et $\forall t \geq d$).

- Pour toute date t de $\mathbb{R}^{+*}$ inférieure à d, la valeur $(f/d)(t)$ de la fonction f/d est le résultat d'un prédicat. Ce prédicat est vrai s'il existe une date t_1 précédant t telle que la valeur $f(t_1)$ du signal f vaut 1.
- Pour toute date t de $\mathbb{R}^{+*}$ supérieure à d, la valeur $(f/d)(t)$ de la fonction f/d est le résultat d'un prédicat. Ce prédicat est vrai s'il existe une date t_1 telle que $t-d < t_1 \leq t$ pour laquelle la valeur $f(t_1)$ du signal f vaut 1.

La figure 20 montre un signal f de $\mathbb{I}$ et le résultat du traitement de ce signal par l'opération TOF.

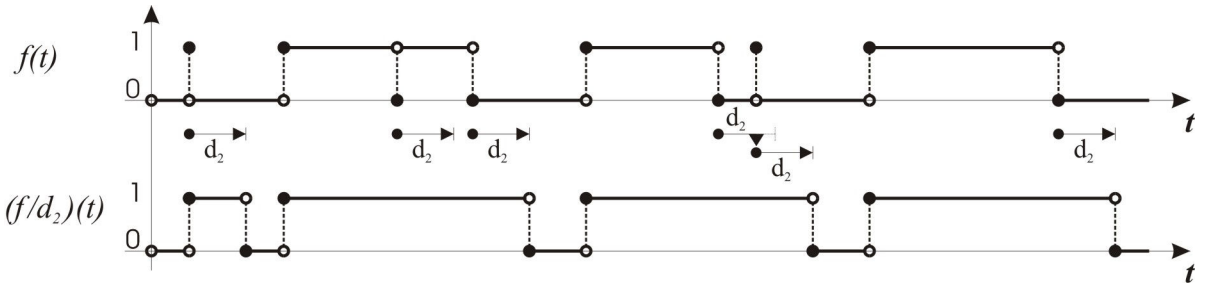


Figure 20 *Représentation graphique de la fonction TOF : f/d_2*

3.4.3 Théorèmes relatifs aux opérations TON et TOF

A partir de la définition mathématique des opérations TON et TOF, il est possible de démontrer les théorèmes suivants :

$$d/f \leq f \qquad [3.41]$$

$$f \leq f/d \qquad [3.42]$$

$$d/(f \cdot g) = (d/f) \cdot (d/g) \qquad [3.43]$$

$$(f+g)/d = f/d + g/d \qquad [3.44]$$

$$\forall d_2 > d_1 \qquad (d_2 / f) \le (d_1 / f) \qquad [3.45]$$

$$\forall d_2 > d_1 \qquad (f / d_1) \le (f / d_2) \qquad [3.46]$$

$$(d_1 / f) \cdot (d_2 / f) = max(d_1, d_2) / f \qquad [3.47]$$

$$(d_1 / f) + (d_2 / f) = min(d_1, d_2) / f \qquad [3.48]$$

$$(f / d_1) + (f / d_2) = f / max(d_1, d_2) \qquad [3.49]$$

$$(f / d_1) \cdot (f / d_2) = f / min(d_1, d_2) \qquad [3.50]$$

$$\forall t \ge d \qquad \overline{(d / f)} = \overline{f} / d \qquad [3.51]$$

$$\forall t \ge d \qquad \overline{(f / d)} = d / \overline{f} \qquad [3.52]$$

3.4.4 Démonstrations des théorèmes relatifs aux opérations TON et TOF

> **Note** : Cette sous-section peut être omise en première lecture

Démonstration de la relation [3.41] : $d / f \le f$

Ce théorème sera démontré en utilisant sa forme équivalente ([2.34] ⇔ [2.37]) :

$(d / f) \cdot \overline{f} = 0*$

$$\forall t : 0 < t < d, \quad \left((d / f) \cdot \overline{f}\right)(t) = (d / f)(t) \wedge \overline{f}(t)$$
$$= 0 \wedge \overline{f}(t)$$
$$= 0$$

$$\forall t \ge d, \qquad \left((d / f) \cdot \overline{f}\right)(t) = (d / f)(t) \wedge \overline{f}(t)$$
$$= \left[\forall t_1 \in]t - d, t] : \left(f(t_1) = 1\right)\right] \wedge \neg f(t)$$
$$= \left(\left[\forall t_1 \in]t - d, t[: \left(f(t_1) = 1\right)\right] \wedge f(t)\right) \wedge \neg f(t)$$
$$= 0$$

En conclusion, nous avons bien $\forall t \in \mathbb{R}^{+*}, \ \left((d / f) \cdot \overline{f}\right)(t) = 0 = 0*(t)$

donc, $(d / f) \cdot \overline{f} = 0*$. D'après l'équivalence [2.37] ⇔ [2.34], si $(d / f) \cdot \overline{f} = 0*$ alors $(d / f) < f$ □

Démonstration de la relation [3.42] : $f \le f / d$

Ce théorème sera démontré en utilisant sa forme équivalente ([2.34] ⇔ [2.35]) :

$f \cdot (f / d) = f$

$$\forall t : 0 < t < d,\ \Big(f \cdot (f/d)\Big)(t) = f(t) \wedge (f/d)(t)$$
$$= f(t) \wedge \Big[\exists t_1 \in]0,t] : f(t_1) = 1\Big]$$
$$= f(t) \wedge \Big(\Big[\exists t_1 \in]0,t[: f(t_1) = 1\Big] \vee f(t)\Big)$$
$$= f(t)$$
$$\forall t \geq d,\quad \Big(f \cdot (f/d)\Big)(t) = f(t) \wedge (f/d)(t)$$
$$= f(t) \wedge \Big[\exists t_1 \in]t-d,t] : \Big(f(t_1) = 1\Big)\Big]$$
$$= f(t) \wedge \Big(\Big[\exists t_1 \in]t-d,t[: \Big(f(t_1) = 1\Big)\Big] \vee f(t)\Big)$$
$$= f(t)$$

En conclusion, nous avons bien $\forall t \in \mathbb{R}^{+*},\ \Big(f \cdot (f/d)\Big)(t) = f(t)$

donc, $f \cdot (f/d) = f$. D'après l'équivalence [2.34] $\Leftrightarrow$ [2.35], si $f \cdot (f/d) = f$ alors $f < (f/d)$ □

Démonstration de l'égalité [3.43] : $d/(f \cdot g) = (d/f) \cdot (d/g)$

$$\forall t : 0 < t < d,\ \Big(d/(f \cdot g)\Big)(t) = 0 = 0 \wedge 0 = (d/f)(t) \wedge (d/g)(t)$$
$$= \Big((d/f) \cdot (d/g)\Big)(t)$$
$$\forall t \geq d,\quad \Big(d/(f \cdot g)\Big)(t) = \Big[\forall t_1 \in]t-d,t] : (f \cdot g)(t_1) = 1\Big]$$
$$= \Big[\forall t_1 \in]t-d,t] : \Big(f(t_1) \wedge g(t_1)\Big) = 1\Big]$$
$$= \Big[\forall t_1 \in]t-d,t] : f(t_1) = 1\Big] \wedge \Big[\forall t_1 \in]t-d,t] : g(t_1) = 1\Big]$$
$$= (d/f)(t) \wedge (d/g)(t)$$
$$= \Big((d/f) \cdot (d/g)\Big)(t)$$

En conclusion, nous avons bien $\forall t \in \mathbb{R}^{+*},\ \Big(d/(f \cdot g)\Big)(t) = \Big((d/f) \cdot (d/g)\Big)(t)$

donc, $d/(f \cdot g) = (d/f) \cdot (d/g)$ □

Démonstration de l'égalité [3.44] : $(f+g)/d = f/d + g/d$

$$\forall t : 0 < t < d,\ \Big((f+g)/d\Big)(t) = \Big[\exists t_1 \in]0,t], (f+g)(t_1) = 1\Big]$$
$$= \Big[\exists t_1 \in]0,t], \Big(f(t_1) \vee g(t_1)\Big) = 1\Big]$$
$$= \Big[\exists t_1 \in]0,t], f(t_1) = 1\Big] \vee \Big[\exists t_1 \in]0,t], g(t_1) = 1\Big]$$
$$= (f/d)(t) \vee (g/d)(t)$$
$$= \Big((f/d) + (g/d)\Big)(t)$$

$$\forall t \geq d, \qquad \Big((f+g)/d\Big)(t) = \Big[\exists t_1 \in]t-d,t], (f+g)(t_1) = 1\Big]$$
$$= \Big[\exists t_1 \in]t-d,t], \Big(f(t_1) \vee g(t_1)\Big) = 1\Big]$$
$$= \Big[\exists t_1 \in]t-d,t], f(t_1) = 1\Big] \vee \Big[\exists t_1 \in]t-d,t], g(t_1) = 1\Big]$$
$$= (f/d)(t) \vee (g/d)(t)$$
$$= \Big((f/d) + (g/d)\Big)(t)$$

En conclusion, nous avons bien $\forall t \in \mathbb{R}^{+*}, \ \Big((f+g)/d\Big)(t) = \Big((f/d) + (g/d)\Big)(t)$

donc, $(f+g)/d = (f/d) + (g/d)$ □

Démonstration de la relation [3.45] : $\forall d_2 > d_1 \qquad (d_2/f) \leq (d_1/f)$

Ce théorème sera démontré en utilisant sa forme équivalente ([2.34] ⇔ [2.35]) :

$(d_1/f) \cdot (d_2/f) = (d_2/f)$

$$\forall t : 0 < t < d_1, \ \Big((d_1/f)\cdot(d_2/f)\Big)(t) = (d_1/f)(t) \wedge (d_2/f)(t) = 0 \wedge 0 = 0$$
$$= (d_2/f)(t)$$
$$\forall t \in [d_1, d_2[, \ \Big((d_1/f)\cdot(d_2/f)\Big)(t) = (d_1/f)(t) \wedge (d_2/f)(t)$$
$$= \Big[\forall t_1 \in]t-d_1,t] : f(t_1) = 1\Big] \wedge 0 = 0$$
$$= (d_2/f)(t)$$
$$\forall t \geq d_2, \quad \Big((d_1/f)\cdot(d_2/f)\Big)(t) = (d_1/f)(t) \wedge (d_2/f)(t)$$
$$= \Big[\forall t_1 \in]t-d_1,t] : f(t_1) = 1\Big] \wedge \Big[\forall t_1 \in]t-d_2,t] : f(t_1) = 1\Big]$$
$$= \Big[\forall t_1 \in]t-d_2,t] : f(t_1) = 1\Big]$$
$$= (d_2/f)(t)$$

En conclusion, nous avons bien $\forall d_2 > d_1, \ \forall t \in \mathbb{R}^{+*}, \ \Big((d_1/f)\cdot(d_2/f)\Big)(t) = (d_2/f)(t)$

donc, $\forall d_2 > d_1, \ (d_1/f)\cdot(d_2/f) = (d_2/f)$ □

Démonstration de la relation [3.46] : $\forall d_2 > d_1 \qquad (f/d_1) \leq (f/d_2)$

Ce théorème sera démontré en utilisant sa forme équivalente ([2.34] ⇔ [2.35]) :

$(f/d_1) \cdot (f/d_2) = (f/d_1)$

$$\forall t : 0 < t < d_1 \ \Big((f/d_1)\cdot(f/d_2)\Big)(t) = (f/d_1)(t) \wedge (f/d_2)(t)$$
$$= \Big[\exists t_1 \in]0,t] : f(t_1) = 1\Big] \wedge \Big[\exists t_1 \in]0,t] : f(t_1) = 1\Big]$$
$$= \Big[\exists t_1 \in]0,t] : f(t_1) = 1\Big]$$
$$= (f/d_1)(t)$$

$$\forall t \in [d_1, d_2[,\ \Big((f/d_1)\cdot(f/d_2)\Big)(t) = (f/d_1)(t) \wedge (f/d_2)(t)$$
$$= \Big[\exists t_1 \in]t-d_1, t] : f(t_1) = 1\Big] \wedge \Big[\exists t_1 \in]0, t] : f(t_1) = 1\Big]$$
$$= \Big[\Big(\exists t_1 \in]t-d_1, t] \wedge \exists t_1 \in]0, t]\Big) : f(t_1) = 1\Big]$$
$$= \Big[\exists t_1 \in]t-d_1, t] : f(t_1) = 1\Big]$$
$$= (f/d_1)(t)$$

$$\forall t \geq d_2,\quad \Big((f/d_1)\cdot(f/d_2)\Big)(t) = (f/d_1)(t) \wedge (f/d_2)(t)$$
$$= \Big[\exists t_1 \in]t-d_1, t] : f(t_1) = 1\Big] \wedge \Big[\exists t_1 \in]t-d_2, t] : f(t_1) = 1\Big]$$
$$= \Big[\Big(\exists t_1 \in]t-d_1, t] \wedge \exists t_1 \in]t-d_2, t]\Big) : f(t_1) = 1\Big]$$
$$= \Big[\exists t_1 \in]t-d_1, t] : f(t_1) = 1\Big]$$
$$= (f/d_1)(t)$$

En conclusion, nous avons bien $\forall d_2 > d_1,\ \forall t \in \mathbb{R}^{+*},\ \Big((f/d_1)\cdot(f/d_2)\Big)(t) = (f/d_1)(t)$

donc, $\forall d_2 > d_1,\quad (f/d_1)\cdot(f/d_2) = (f/d_1)$ □

Démonstration de l'égalité [3.47] : $(d_1/f)\cdot(d_2/f) = max(d_1, d_2)/f$

Pour réaliser cette démonstration, il est nécessaire de distinguer les trois cas suivants :

$d_1 = d_2$ $\qquad$ $d_1 < d_2$ $\qquad$ $d_1 > d_2$

- Premier cas : $d_1 = d_2$

  Si $d_1 = d_2 = d$, alors $max(d_1, d_2) = d$.

  En reportant ce résultat, nous avons :

  $(d_1/f)\cdot(d_2/f) = (d/f)\cdot(d/f) = (d/f) = max(d_1, d_2)/f$

- Deuxième cas : $d_1 < d_2$

  Si $d_1 < d_2$, alors $max(d_1, d_2) = d_2$.

  Si $d_1 < d_2$, alors $(d_2/f) \leq (d_1/f)$ (d'après [3.45])

  $(d_2/f) \leq (d_1/f) \Leftrightarrow (d_1/f)\cdot(d_2/f) = (d_2/f)$ (en utilisant [2.34] ⇔ [2.35])

  $= max(d_1, d_2)/f$

- Troisième cas : $d_1 > d_2$

  Si $d_1 > d_2$, alors $max(d_1, d_2) = d_1$.

  Si $d_1 > d_2$, alors $(d_1/f) \leq (d_2/f)$ (d'après [3.45])

$(d_1 / f) \le (d_2 / f) \Leftrightarrow (d_1 / f) \cdot (d_2 / f) = (d_1 / f)$ (en utilisant [2.34] ⇔ [2.35])

$= max(d_1, d_2) / f$

En conclusion, nous avons bien

$(d_1 / f) \cdot (d_2 / f) = max(d_1, d_2) / f$ □

Démonstration de l'égalité [3.48] : $(d_1 / f) + (d_2 / f) = min(d_1, d_2) / f$

Pour réaliser cette démonstration, il est nécessaire de distinguer les trois cas suivants :

$d_1 = d_2$ $d_1 < d_2$ $d_1 > d_2$

- Premier cas : $d_1 = d_2$

  Si $d_1 = d_2 = d$, alors $min(d_1, d_2) = d$.

  En reportant ce résultat, nous avons :

  $(d_1 / f) + (d_2 / f) = (d / f) + (d / f) = (d / f) = min(d_1, d_2) / f$

- Deuxième cas : $d_1 < d_2$

  Si $d_1 < d_2$, alors $min(d_1, d_2) = d_1$.

  Si $d_1 < d_2$, alors $(d_2 / f) \le (d_1 / f)$ (d'après [3.45])

  $(d_2 / f) \le (d_1 / f) \Leftrightarrow (d_1 / f) + (d_2 / f) = (d_1 / f)$ (en utilisant [2.34] ⇔ [2.39])

  $= min(d_1, d_2) / f$

- Troisième cas : $d_1 > d_2$

  Si $d_1 > d_2$, alors $min(d_1, d_2) = d_2$.

  Si $d_1 > d_2$, alors $(d_1 / f) \le (d_2 / f)$ (d'après [3.45])

  $(d_1 / f) \le (d_2 / f) \Leftrightarrow (d_1 / f) + (d_2 / f) = (d_2 / f)$ (en utilisant [2.34] ⇔ [2.39])

  $= min(d_1, d_2) / f$

En conclusion, nous avons bien

$(d_1 / f) + (d_2 / f) = min(d_1, d_2) / f$ □

Démonstration de l'égalité [3.49] : $(f / d_1) + (f / d_2) = f / max(d_1, d_2)$

Pour réaliser cette démonstration, il est nécessaire de distinguer les trois cas suivants :

$d_1 = d_2$ $d_1 < d_2$ $d_1 > d_2$

- Premier cas : $d_1 = d_2$

  Si $d_1 = d_2 = d$, alors $max(d_1, d_2) = d$.

En reportant ce résultat, nous avons :

$(f/d_1)+(f/d_2)=(f/d)+(f/d)=(f/d)=f/max(d_1,d_2)$

- Deuxième cas : $d_1 < d_2$

  Si $d_1 < d_2$, alors $max(d_1,d_2)=d_2$.

  Si $d_1 < d_2$, alors $(f/d_1) \le (f/d_2)$ (d'après [3.46])

  $(f/d_1) \le (f/d_2) \Leftrightarrow (f/d_1)+(f/d_2)=(f/d_2)$ (en utilisant [2.34] ⇔ [2.39])

  $= f/max(d_1,d_2)$

- Troisième cas : $d_1 > d_2$

  Si $d_1 > d_2$, alors $max(d_1,d_2)=d_1$.

  Si $d_1 > d_2$, alors $(f/d_2) \le (f/d_1)$ (d'après [3.46])

  $(f/d_2) \le (f/d_1) \Leftrightarrow (f/d_1)+(f/d_2)=(f/d_1)$ (en utilisant [2.34] ⇔ [2.39])

  $= f/max(d_1,d_2)$

En conclusion, nous avons bien

$(f/d_1)+(f/d_2)=f/max(d_1,d_2)$ □

Démonstration de l'égalité [3.50] : $(f/d_1)\cdot(f/d_2)=f/min(d_1,d_2)$

Pour réaliser cette démonstration, il est nécessaire de distinguer les trois cas suivants :

$d_1 = d_2$ $d_1 < d_2$ $d_1 > d_2$

- Premier cas : $d_1 = d_2$

  Si $d_1 = d_2 = d$, alors $min(d_1,d_2)=d$.

  En reportant ce résultat, nous avons :

  $(f/d_1)\cdot(f/d_2)=(f/d)\cdot(f/d)=(f/d)=f/min(d_1,d_2)$

- Deuxième cas : $d_1 < d_2$

  Si $d_1 < d_2$, alors $min(d_1,d_2)=d_1$.

  Si $d_1 < d_2$, alors $(f/d_1) \le (f/d_2)$ (d'après [3.46])

  $(f/d_1) \le (f/d_2) \Leftrightarrow (f/d_1)\cdot(f/d_2)=(f/d_1)$ (en utilisant [2.34] ⇔ [2.35])

  $= f/min(d_1,d_2)$

- Troisième cas : $d_1 > d_2$

  Si $d_1 > d_2$, alors $min(d_1,d_2)=d_2$.

Si $d_1 > d_2$, alors $(f/d_2) \le (f/d_1)$ (d'après [3.46])

$(f/d_2) \le (f/d_1) \quad \Leftrightarrow \quad (f/d_1)\cdot(f/d_2) = (f/d_2)$ (en utilisant [2.34] $\Leftrightarrow$ [2.35])

$= f/min(d_1, d_2)$

En conclusion, nous avons bien

$(f/d_1)\cdot(f/d_2) = f/min(d_1, d_2)$ □

Démonstration de l'égalité [3.51] : $\forall t \ge d \qquad \overline{(d/f)} = \overline{f}/d$

$$\begin{aligned} \forall t \ge d, \quad \overline{(d/f)}(t) &= \overline{\left(\left[\forall t_1 \in]t-d, t], f(t_1) = 1\right]\right)} \\ &= \left[\exists t_1 \in]t-d, t], f(t_1) = 0\right] \\ &= \left[\exists t_1 \in]t-d, t], \overline{f}(t_1) = 1\right] \\ &= (\overline{f}/d)(t) \end{aligned}$$ □

Démonstration de l'égalité [3.52] : $\forall t \ge d \qquad \overline{(f/d)} = d/\overline{f}$

$\forall t \ge d, \quad \left(\overline{f/d}\right) = \left(\overline{\overline{f}/d}\right) = \left(\overline{d/\overline{f}}\right) = d/\overline{f}$ (en utilisant [2.18] et[3.51])

3.5 Conclusion

Nous avons présenté dans ce chapitre les opérations de l'algèbre 𝕀 permettant de décrire des événements et des comportements séquentiels et temporisés. Ces opérations complètent le cadre formel décrit au chapitre 2 et que nous utiliserons dans les chapitres suivants pour formaliser les spécifications de la commande d'un système logique et générer les lois de commande à partir de ces spécifications.

Chapitre 4

Nous montrerons dans ce chapitre comment il est possible de spécifier formellement le comportement d'un contrôleur logique à l'aide de relations de l'algèbre Ⅱ.

Après avoir défini quatre relations types, trois exemples de complexité croissante illustrent l'approche.

4. Traduction de spécifications informelles en un ensemble de relations sur 𝕀

L'objectif de ce chapitre (figure 21) est de développer la première étape de notre méthode de synthèse, à savoir la traduction de spécifications informelles en spécifications formelles, sous forme de relations de l'algèbre 𝕀.

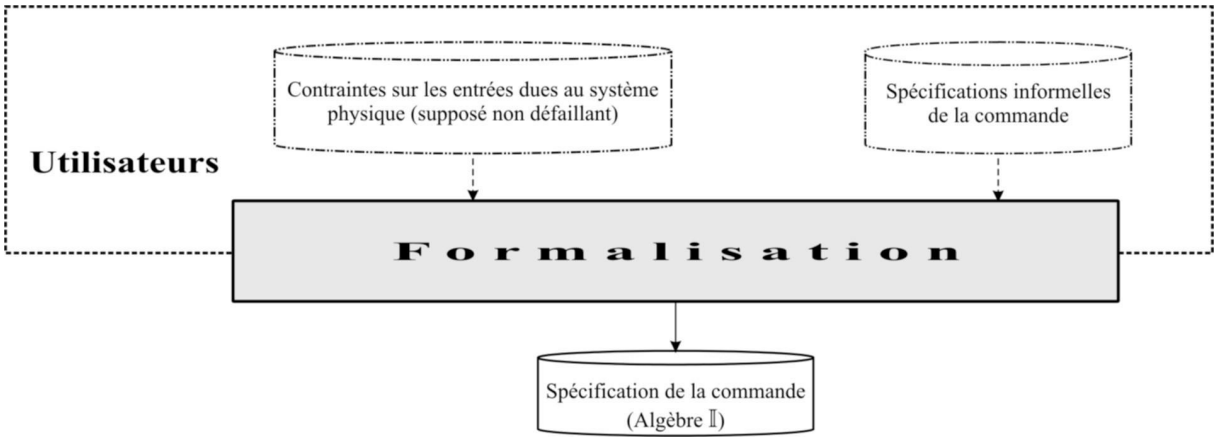


Figure 21 *Objectif du chapitre : Formalisation de la commande en algèbre 𝕀*

Nous soulignons qu'outre l'ensemble des spécifications informelles traduisant le fonctionnement normal et les contraintes de sécurité, cette étape de formalisation peut accepter des contraintes sur les entrées du contrôleur logique, (généralement sous forme d'une relation d'égalité dont l'un de membres est 1* et 0*) dues au système physique, supposé non défaillant. Ces contraintes traduisent des comportements normaux observés par les détecteurs, comme par exemple la non-simultanéité des valeurs vraies de deux détecteurs fin de course.

Dans un premier temps nous proposons une classification des spécifications formelles que nous utiliserons, puis nous illustrerons la méthode de formalisation des spécifications à l'aide de trois exemples de complexité croissante (tableau ci-dessous).

Exemple	Caractéristiques
1) Commande du remplissage d'un réservoir	Système mono-sortie
2) Commande d'un portail automatique	Système à deux sorties avec contraintes sur les entrées et les sorties
3) Commande d'un préhenseur effectuant un cycle en U	Système à plusieurs sorties avec contraintes sur les entrées et les sorties

4.1 Classification des spécifications

Rappelons que le modèle de base de notre approche est un système de commande logique à *m* entrées et *n* sorties, qui peut être implanté dans un API.

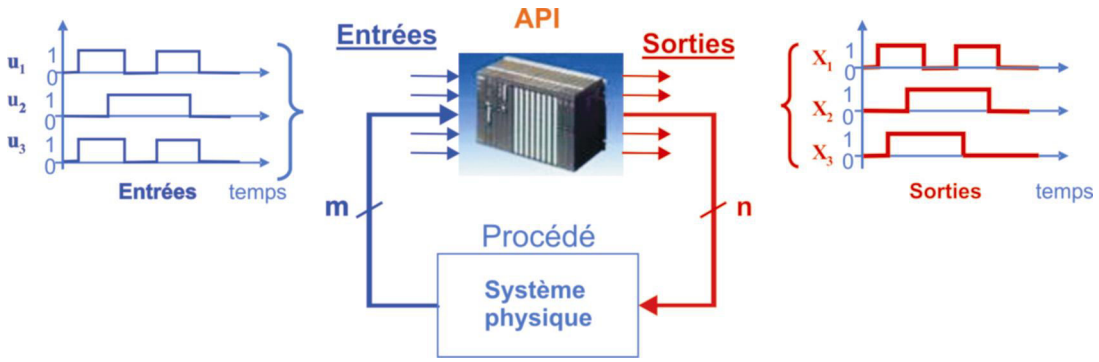
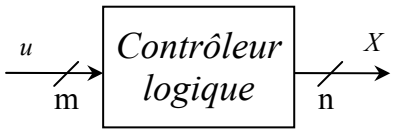


Figure 22 *Modèle du système de contrôle-commande*

Du point de vue de l'algèbre $\mathbb{I}$, un contrôleur logique est donc un système qui reçoit des signaux d'entrée et génère des signaux de sortie (figure 23).



Nous noterons : u : l'ensemble des entrées
u_i : le i ème élément de cet ensemble
X : l'ensemble des sorties
X_j : le j ème élément de cet ensemble

Figure 23 *Contrôleur logique*

De plus, nous rappelons que nous avons défini des opérations (ET, OU, NON, RE, FE, RS, SR, TON et TOF) et deux relations (= et ≤) sur l'algèbre $\mathbb{I}$.

Une spécification formelle est alors une relation (= ou ≤) entre deux signaux qui peuvent être :

- Un signal d'entrée ou un signal de sortie

- Un signal obtenu par combinaison, à l'aide des opérations définies, de signaux d'entrée et/ou de sortie
- Les signaux 1* ou 0*

Plus précisément, nous nous limitons à quatre types de spécifications :

Classification	**Représentation générale formelle en** $\mathbb{I}$	
SF-T1	$F(u) \leq X_i$	Où : $F(u)$ est une fonction d'entrées $F(X)$ est une fonction de sorties $F(u,X)$ est une fonction d'entrées et de sorties
SF-T2	$F_1(u,X) \leq F_2(X)$	
SF-T3	$F(u) = 0*$	
SF-T4	$F(X) = 0*$	

La première spécification (**SF-T1**) est une relation qui définit un signal de sortie à partir de signaux d'entrée. Cette relation indique en effet que pour toutes les dates où $F(u)$ vaut 1, X_i doit valoir 1[1].

La seconde spécification (**SF-T2**) est une relation, plus complexe, entre une fonction d'entrées et de sorties et une fonction de sorties.

Enfin, les deux dernières spécifications sont des relations qui expriment des contraintes entre signaux d'entrée (**SF-T3** : hypothèses de bon fonctionnement de la partie opérative) et signaux de sortie (**SF-T4** : contraintes de sécurité en général).

Nous supposerons dans tout ce qui suit qu'il est toujours possible de mettre sous la forme d'un ensemble de relations de ces quatre types les spécifications d'un contrôleur logique. Cette hypothèse nous paraît raisonnable comme le montrent les exemples présentés à la section trois.

4.2 Introduction à la formalisation des spécifications

Nous allons considérer les propositions données dans la spécification de fonctionnement (SF) comme des assertions. Une fois formalisées en $\mathbb{I}$, ces assertions ne seront remises en cause que si elles s'avèrent incohérentes les unes par rapport aux autres :

> Une ***assertion*** est une proposition qui, dans sa forme, peut être affirmative ou négative, que l'on avance et que l'on soutient comme vraie[2].

[1] On peut noter que ce type de spécification n'impose aucune contrainte sur la valeur de X_i quand la valeur de $F(u)$ vaut 0.

[2] Définition proposée par Le Petit Robert

Il existe deux éléments et une relation dans l'algèbre $\mathbb{I}$, qui jouent un rôle prépondérant. Ce sont les élément 1* (toujours), 0* (jamais) et la relation d'ordre partiel "≤".
La réussite de la formalisation complète de la spécification passe donc pour la bonne compréhension de la relation d'ordre partiel (≤) et des deux éléments (1*, 0*) qui ont été introduits dans l'algèbre $\mathbb{I}$.

Toujours : (1*)

L'élément 1* est utilisé pour représenter des assertions qui expriment des propriétés qui doivent être toujours vérifiées. Cet élément (1*) est utilisé dans la SF car les propriétés à respecter sont considérées aussi comme assertions : A_1, A_2, A_3, … etc.

Ainsi, étant donné que toutes les propriétés de la SF doivent être respectées, alors le but de notre méthode est de garantir que : $(A_1 \cdot A_2 \cdot A_3 \cdot \quad) = 1^*$

A titre d'exemple d'énoncés en langage naturel (LN) équivalents à $f + g = 1^*$, nous pouvons citer

- À chaque instant, au moins une des valeurs des signaux f et g est vraie.
- Le signal f ou g arrive TOUJOURS.

Jamais : (0*)

Cet élément peut être utilisé aussi pour définir des assertions. En effet, le complément d'une assertion qui est toujours vraie est une assertion toujours fausse.
A titre d'exemple nous pouvons citer que $f \cdot g = 0^*$ peut provenir des énoncés en LN suivants (noter que $f \cdot g = 0^* \Leftrightarrow \overline{f} + \overline{g} = 1^*$ car [2.54] ⇔ [2.56]) :

- Le signal f et le signal g ne sont JAMAIS vrais en même temps.
- Les valeurs des signaux f et g ne sont JAMAIS simultanément vraies.

Relation d'ordre partiel (Inclusion) : ≤

L'opération inclusion permet de traduire des énoncés du type **SI … ALORS …**. Cependant il faut prendre garde de ne pas interpréter ces énoncés comme des relations de cause à conséquence avec un décalage temporel. Nous considérons en effet le système de commande infiniment réactif, la cause et la conséquence se produisant à la même date. L'inclusion $f \leq g$ signifie donc "La valeur du signal f vaut 1 uniquement lorsque celle du signal g est égale à 1".

Nous pouvons citer d'autres expressions équivalentes :

- **SI** f est vrai 1, **ALORS** g vaut 1.
- Il est suffisant que la valeur du signal f soit vraie pour que celle du signal g le soit.
- Quand la valeur du signal f est vraie, alors celle du signal g l'est aussi.

4.3 Illustration de la formalisation à l'aide de trois exemples

Nous allons présenter dans cette section la première partie de la méthode d'élaboration de la commande (figure 21) à l'aide de trois exemples. En utilisant ces trois exemples, nous allons illustrer le passage d'une spécification informelle de la commande désirée et des contraintes sur les entrées, décrites en LN, à la spécification formelle sur $\mathbb{I}$.

4.3.1 Exemple 1 : Système à une sortie

L'exemple traité dans cette section concerne un système à une sortie. En utilisant cet exemple nous allons introduire la formalisation des spécifications de type SF-T1.

4.3.1.1 Description du système

Le système étudié dans cette section (voir figure 24a) est la commande du remplissage d'un réservoir à partir d'un bassin. Nous considérerons que le remplissage se fait par l'intermédiaire d'une pompe entraînée par un moteur électrique à vitesse unique. Le réservoir est équipé de deux capteurs permettant de connaître son niveau tandis que le bassin n'est équipé que d'un seul capteur. Les spécifications en langage naturel du fonctionnement souhaité sont données dans la figure 24d. Nous remarquons que l'assertion A3 a été incluse afin de ne pas pomper à vide et éviter le risque d'endommager la pompe.

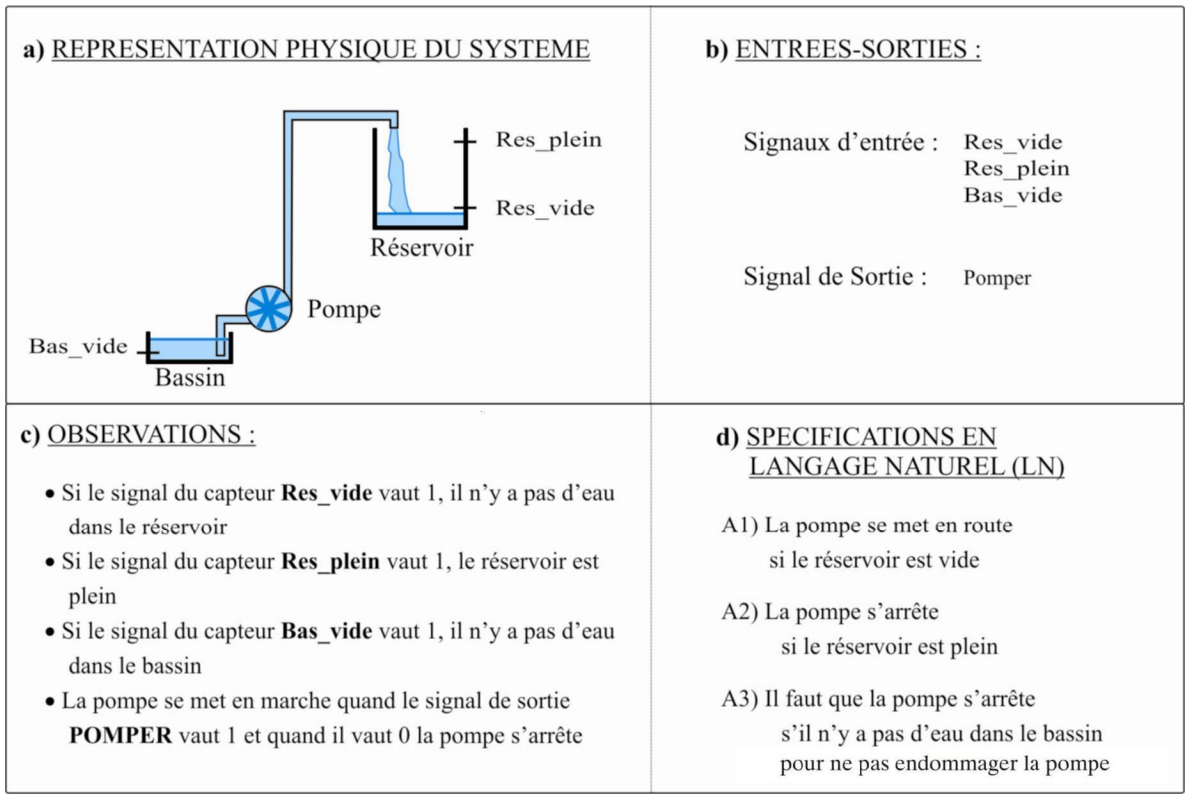


Figure 24 *Présentation de l'exemple 1*

4.3.1.2 Formalisation des spécifications

Les signaux d'entrée sont : Res_vide , Res_plein et Bas_vide , le signal de sortie est $Pomper$ (voir figure 24b et 4c) . Nous réécrirons d'abord les spécifications sous forme d'expressions SI, … ALORS, puis nous donnerons la représentation formelle sur $\mathbb{I}$ de ces expressions.

Relations données en langage naturel (LN)	**Formalisation sur $\mathbb{I}$**
A1) **SI** le signal "réservoir vide " vaut 1, **ALORS** le signal "pomper" vaut 1.	$Res\_vide \leq Pomper$
A2) **SI** le signal "réservoir plein" vaut 1, **ALORS** le signal "pomper " vaut 0.	$Res\_plein \leq \overline{Pomper}$
A3) **SI** le signal "bassin vide" vaut 1, **ALORS** il faut que le signal "pomper" vaut 0.	$Bas\_vide \leq \overline{Pomper}$

Sur le plan mathématique, cet ensemble de relations entre signaux binaires, peut être modélisé par un système d'équations sur $\mathbb{I}$ pour lequel les données sont les signaux Res_vide , Res_plein et Bas_vide et l'inconnue est le signal $Pomper$.

La représentation formelle de ce système est donnée par la figure 25. Notons que ces assertions sont toutes des spécifications de type SF-T1.

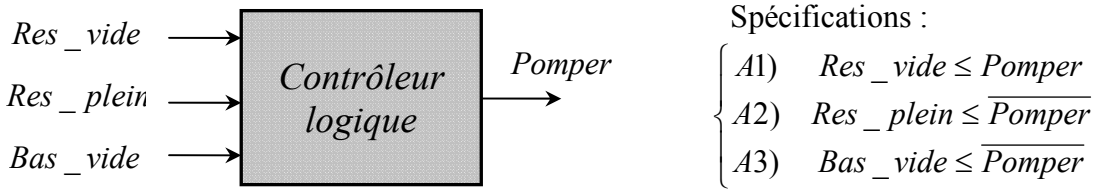


Figure 25 *Représentation formelle de l'exemple 1*

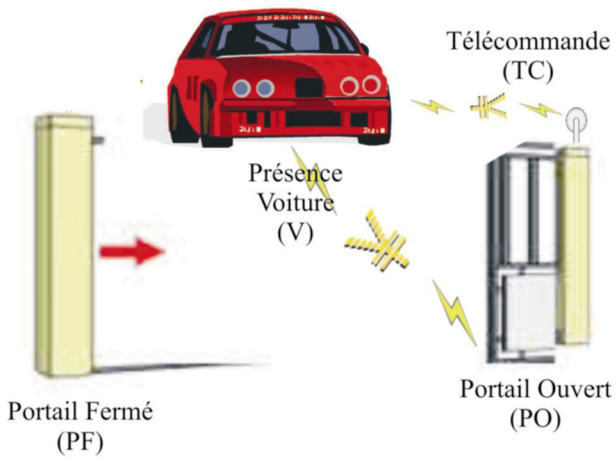
4.3.2 Exemple 2 : Système à deux sorties avec des contraintes sur les sorties

L'exemple traité dans cette section concerne un système à deux sorties. En utilisant cet exemple nous allons formaliser des spécifications de types SF-T1, SF-T3 et SF-T4.

4.3.2.1 Description du système

Le système étudié dans cette section (voir figure 26a) est la commande d'un portail automatique. Nous considérerons que l'ouverture et fermeture se font par l'intermédiaire d'un moteur à deux sens de rotation. Il est considéré qu'il existe une télécommande qui demande l'ouverture du portail, un capteur qui détecte la présence d'une voiture et deux capteurs permettant de connaître si le portail est complètement ouvert ou fermé. Les spécifications en langage naturel du fonctionnement souhaité sont données dans la figure 26d. Nous incluons dans la SF de cet exemple une contrainte sur les sorties (A4) et une contrainte due au système physique (A7).

a) REPRESENTATION PHYSIQUE DU SYSTEME



b) ENTREES-SORTIES :

Signaux d'entrée : PO, PF, TC et V

Signaux de sortie : Ouvrir et Fermer

c) OBSERVATIONS :

- Si les signaux des capteurs **PO** et **PF** valent 1, le portail est ouvert et fermé, respectivement.
- Les signaux des capteurs **TC** et **V** à 1 indiquent que la télécommande est activée et qu'il détecté la présence d'une voiture, respectivement.
- Le moteur pour ouvrir (fermer) le portail est commandé quand le signal de sortie **OUVRIR (FERMER)** vaut 1.

d) SPECIFICATIONS EN LANGAGE NATUREL (LN)

A1) Quand la télécommande est activée, le portail s'ouvre.

A2) Lorsque le portail est ouvert depuis 3 secondes sans une demande de l'utilisateur ou une détection de voiture, le portail se referme.

A3) Tant que le portail n'est pas fermé, la détection d'une voiture provoque son ouverture.

A4) Le portail ne doit jamais être contrôlé simultanément en ouverture et en fermeture.

A5) Ne pas commander l'ouverture si le portail est ouvert.

A6) Ne pas commander la fermeture si le portail est fermé.

A7) Le portail n'est jamais simultanément en position ouverte et en position fermée.

Figure 26 *Présentation de l'exemple 2*

4.3.2.2 Formalisation

Les signaux d'entrée sont : TC , PO , PF et V , les signaux de sortie sont *Ouvrir* et *Fermer* (voir figure 26) . Nous réécrirons d'abord les spécifications sous forme d'expressions SI, … ALORS et JAMAIS, puis nous donnerons la représentation formelle sur $\mathbb{I}$ de ces expressions dans la figure 27. De plus, la spécification A2 nous amène à introduire l'opération TON.

Relations données en langage naturel (LN)

A1) **SI** le signal "télécommande " vaut 1, **ALORS** le signal "Ouvrir" vaut 1.

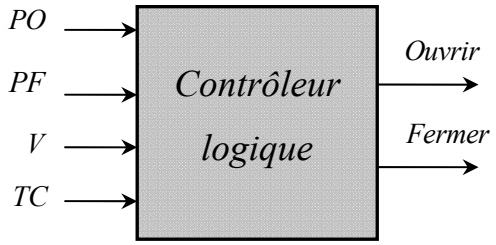
A2) **SI** le signal "depuis 3 secondes le portail est ouvert **sans** l'activation de la télécommande par l'utilisateur ou une détection de voiture" vaut 1, **ALORS** le signal "Fermer" vaut 1.

A3) **SI** le signal "le portail n'est pas fermé et la détection d'une voiture" vaut 1, **ALORS** il faut que le signal " Ouvrir " soit à 1.

A4) Le signal "ouvrir et fermer" ne doit **JAMAIS** valoir 1.
A5) **SI** le signal "portail ouvert" vaut 1, **ALORS** le signal "Ouvrir" vaut 0.
A6) **SI** le signal "portail fermé" vaut 1, **ALORS** le signal "Fermer" vaut 0.
A7) Le signal " portail ouvert et portail fermé" ne doit **JAMAIS** valoir 1.

Ainsi, sur le plan mathématique, cet ensemble de relations entre signaux binaires, peut être modélisé par un système d'équations sur $\mathbb{I}$ pour lequel les données sont les signaux TC, PO, PF et V, et les inconnues sont les signaux $Ouvrir$ et $Fermer$. La formalisation de ces spécifications nous amènent à introduire l'élément 0* (A4 et A7) et l'opération TON (A2).

La représentation formelle de ce système est donnée par la figure 27. Notons que les assertions A1, A2, A3, A5 et A6 sont des spécifications de type SF-T1, l'assertion A7 est de type SF-T3 et l'assertion A4 est de type SF-T4.



Spécifications formelles sur $\mathbb{I}$:

$$\begin{cases} A1) & TC \le Ouvrir \\ A2) & 3s/(PO \cdot \overline{TC} \cdot \overline{V}) \le Fermer \\ A3) & \overline{PF} \cdot V \le Ouvrir \\ A4) & Ouvrir \cdot Fermer = 0* \\ A5) & PO \le \overline{Ouvrir} \\ A6) & PF \le \overline{Fermer} \\ A7) & PO \cdot PF = 0* \end{cases}$$

Figure 27 *Représentation formelle de l'exemple 2*

4.3.3 Exemple 3 : Système à plus de deux sorties

L'exemple traité dans cette section concerne un système à plusieurs sorties. En utilisant cet exemple nous allons formaliser des spécifications de types SF-T1, SF-T2, SF-T3 et SF-T4.

4.3.3.1 Description du système

Le système étudié dans cette section (voir figure 28a) est la commande automatique d'un préhenseur effectuant un cycle en U, afin de prendre une pièce au poste de prise et de la déposer au poste de dépose. Le système est composé de deux vérins : le premier (a) est un vérin simple effet permettant la descente[3] du préhenseur et le deuxième (b) est un vérin double effet permettant l'avancement et le recul du préhenseur. La pièce est prise par un système d'aspiration (c). Le début du cycle est commandé par l'utilisateur en appuyant sur le

[3] La montée est faite en mettant la commande Descendre à 0, car le vérin est de type simple effet.

bouton *dcy* . Il existe un autre bouton permettant d'arrêter le cycle en cas d'urgence (*ARU*). De plus, il existe 4 capteurs de position (voir figure 28a).

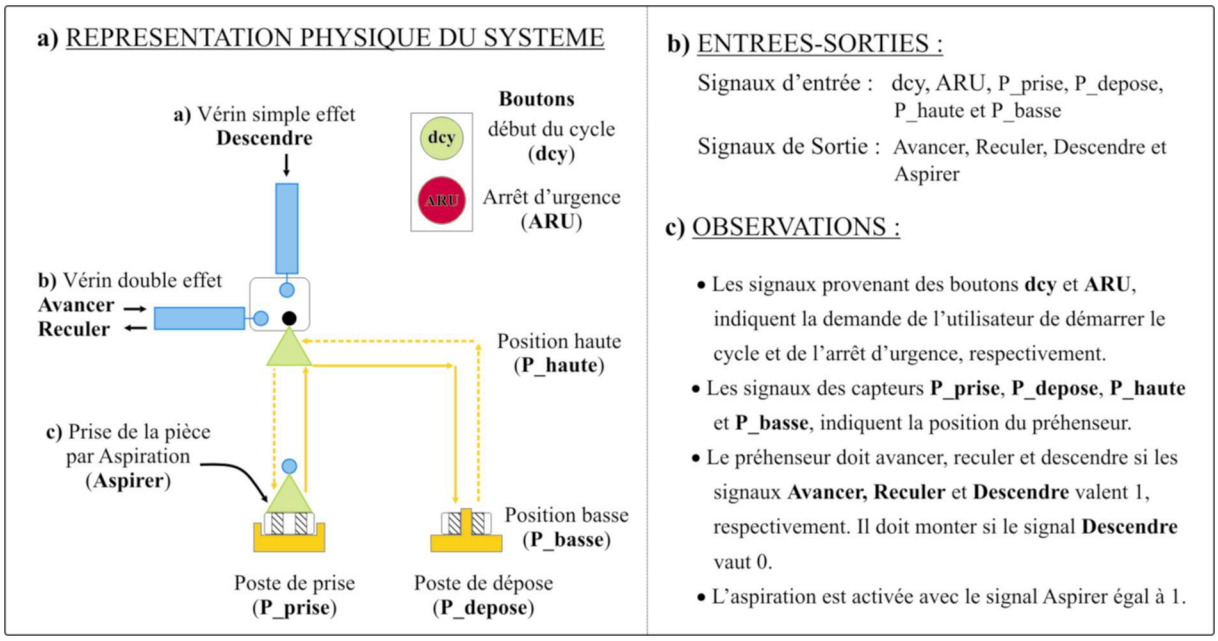


Figure 28 *Présentation de l'exemple 3*

Spécification de la commande du cycle en U (voir figure 28a) :

A1) Suite à l'appui sur le bouton "dcy", si le préhenseur est en position haute et au poste de prise, le préhenseur **descend**. Le cycle en U commence.

A2) Lorsque le préhenseur arrive en position basse, toujours au poste de prise, **aspirer** pour prendre la pièce.

A3) **Arrêter la commande de la descente** après une seconde[4] en position basse et au poste de prise, afin de remonter avec la pièce.

A4) Une fois en position haute et au poste de prise, **avancer** le préhenseur.

A5) En arrivant au poste de dépose, toujours en position haute, **arrêter d'avancer** et **descendre** le préhenseur.

A6) Lorsque le préhenseur arrive en position basse, toujours au poste de dépose, **arrêter d'aspirer** (pour lâcher la pièce) et **arrêter la commande de la descente**.

A7) En étant toujours au poste de dépose, une fois en position haute, **reculer le préhenseur jusqu'au** point de départ (au poste de prise et en position haute).

A8) L'appui sur le bouton d'urgence provoque l'arrêt des commandes : avancer, reculer et descendre.

Le cycle est fini lorsque le préhenseur arrive au poste de prise et, en conséquence, un nouveau cycle peut commencer avec un nouvel appui sur le bouton "dcy".

[4] Ceci afin d'assurer que la pièce a été prise après une seconde d'aspiration (voir A2).

Information sur le système physique :

A9) Les capteurs détectant les positions haute et basse du préhenseur ne fournissent jamais des valeurs vraies simultanément.

A10) Les capteurs détectant les positions du préhenseur au poste de prise et au poste de dépose ne fournissent jamais des valeurs vraies simultanément.

Spécifications de sécurité :

A11) Ne jamais avancer si le préhenseur n'est pas en position haute.

A12) Ne jamais reculer si le préhenseur n'est pas en position haute.

A13) Ne jamais descendre si le préhenseur n'est ni au poste de prise ni au poste de dépose.

A14) Ne jamais demander d'avancer et de reculer en même temps.

A15) Ne jamais demander d'avancer et de descendre en même temps.

A16) Ne jamais demander de reculer et de descendre en même temps.

A11, A12, A13, A15 et A16 sont introduites pour éviter des collisions avec l'environnement.

4.3.3.2 Formalisation

Les signaux d'entrée sont : dcy , ARU , P_prise , P_depose , P_haute et P_basse , les signaux de sortie sont $Avancer$, $Reculer$, $Descendre$ et $Aspirer$ (voir figure 28) . La figure suivante résume la représentation des entrées/sorties de cet exemple, pour lequel les données sont les signaux d'entrée et les inconnus sont les signaux de sortie.

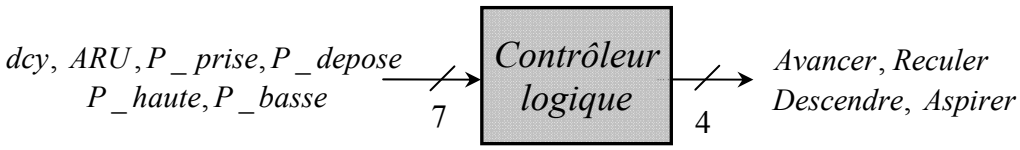


Figure 29 *Entrées/sorties de l'exemple 3*

En utilisant le même principe de réécriture des spécifications sous forme d'expressions SI, … ALORS et JAMAIS, nous donnerons ensuite la représentation formelle sur $\mathbb{I}$ de ces expressions. En effet, sur le plan mathématique, cet ensemble de relations entre signaux binaires peut être modélisé par un système d'équations sur $\mathbb{I}$. Notons que la formalisation de ces spécifications nous amène à introduire l'opération front montant (A1). L'ensemble des spécifications formelles est donc :

Spécification du cycle en U :

$$\begin{cases} A1) & \left(P\_prise \cdot P\_haute \cdot \uparrow dcy\right) \le Descendre \\ A2) & P\_prise \cdot P\_basse \le Aspirer \\ A3) & 1s/(P\_prise \cdot P\_basse) \le \overline{Descendre} \\ A4) & P\_prise \cdot P\_haute \le Avancer \\ A5) & P\_depose \cdot P\_haute \le \overline{Avancer} \cdot Descendre \\ A6) & P\_depose \cdot P\_basse \le \overline{Descendre} \cdot \overline{Aspirer} \\ A7a) & P\_depose \cdot P\_haute \le Reculer \\ A7b) & P\_prise \cdot P\_haute \le \overline{Reculer} \\ A8) & ARU \le \overline{Avancer} \cdot \overline{Reculer} \cdot \overline{Descendre} \end{cases}$$

Spécifications dues aux contraintes sur les entrées :

$$\begin{cases} A9) & P\_prise \cdot P\_depose = 0^* \\ A10) & P\_haute \cdot P\_basse = 0^* \end{cases}$$

Spécifications de sécurité :

$$\begin{cases} A11) & \overline{P\_haute} \cdot Avancer = 0^* \\ A12) & \overline{P\_haute} \cdot Reculer = 0^* \\ A13) & \overline{P\_prise} \cdot \overline{P\_depose} \cdot Descendre = 0^* \\ A14) & Avancer \cdot Reculer = 0^* \\ A15) & Avancer \cdot Descendre = 0^* \\ A16) & Reculer \cdot Descendre = 0^* \end{cases}$$

REMARQUES :

1) Les assertions A11, A12 et A13 sont de type **SF-T1**, de même que A1, A2, A3, A4, A7a et A7b d'ailleurs, car ce sont des assertions avec une seule variable de sortie. En effet, en utilisant [2.34] ⇔ [2.37], nous avons :

$$\begin{cases} A11) & \overline{P\_haute} \cdot Avancer = 0^* & \Leftrightarrow & \overline{P\_haute} \le \overline{Avancer} \\ A12) & \overline{P\_haute} \cdot \mathrm{Re}culer = 0^* & \Leftrightarrow & \overline{P\_haute} \le \overline{\mathrm{Re}culer} \\ A13) & \overline{P\_prise} \cdot \overline{P\_depose} \cdot Descendre = 0^* & \Leftrightarrow & \overline{P\_prise} \cdot \overline{P\_depose} \le \overline{Descendre} \end{cases}$$

2) Les assertions A5, A6 et A8 sont de type **SF-T2**. Nous remarquons cependant que ces assertions peuvent aussi être décomposées en **SF-T1** en utilisant [2.47] :

$$\left\{A5)\quad P\_depose \cdot P\_haute \le \overline{Avancer} \cdot Descendre\right. \quad \Leftrightarrow \quad \begin{cases} A5a) & P\_depose \cdot P\_haute \le \overline{Avancer} \\ A5b) & P\_depose \cdot P\_haute \le Descendre \end{cases}$$

$$\left\{A6)\quad P\_depose \cdot P\_basse \le \overline{Descendre} \cdot \overline{Aspirer}\right. \quad \Leftrightarrow \quad \begin{cases} A6a) & P\_depose \cdot P\_basse \le \overline{Descendre} \\ A6b & P\_depose \cdot P\_basse \le \overline{Aspirer} \end{cases}$$

$$\left\{A8)\quad ARU \le \overline{Avancer} \cdot \overline{Reculer} \cdot \overline{Descendre}\right. \quad \Leftrightarrow \quad \begin{cases} A8a) & ARU \le \overline{Avancer} \\ A8b) & ARU \le \overline{Reculer} \\ A8c) & ARU \le \overline{Descendre} \end{cases}$$

3) Les assertions A9 et A10 sont des **SF-T3** qui indiquent des contraintes du système physique en considérant la non défaillance de capteurs.

4) Les assertions A14, A15 et A16 sont des **SF-T4** qui indiquent des contraintes sur les sorties.

4.4 Conclusion

Nous avons présenté dans ce chapitre quatre types de spécifications (inclusion d'une fonction des signaux d'entrée dans un signal de sortie, inclusion d'une fonction des signaux d'entrée et de signaux de sortie dans une fonction des signaux de sortie, contraintes sur les entrées et contraintes sur les sorties) que nous utilisons pour exprimer formellement le comportement attendu d'un contrôleur logique. A partir de l'ensemble de ces spécifications formelles il est alors possible de trouver les expressions des signaux de sortie en se basant sur les résultats théoriques relatifs à la résolution d'un système d'équations dans l'algèbre $\mathbb{I}$ qui sont présentés au chapitre suivant.

Une nouvelle opération sera définie dans le chapitre suivant. Cette opération est nécessaire pour définir le dernier type de spécification qui sera présenté aussi au prochain chapitre.

Chapitre 5

Après avoir détaillé la manière de passer du cahier de charges d'un système logique séquentiel au modèle algébrique sur $\mathbb{I}$ de sa commande (chapitre 4) nous allons présenter dans ce chapitre la résolution d'un système d'équations particulier sur $\mathbb{I}$.

Ce système, composé de 2 relations d'inclusion, peut être facilement déduit des spécifications formelles.

Dans un premier temps, nous donnerons les conditions de cohérence et de complétude de ce système. Nous nous attacherons ensuite à la résolution d'un système incomplètement spécifié, en proposant une solution générale, puis une solution particulière.

5. Résolution d'un système d'équations

5.1 Introduction

Nous nous intéressons dans ce chapitre à la résolution dans l'algèbre $\mathbb{I}$ d'un système d'équations de la forme :

$$\begin{cases} f \le X \\ g \le \overline{X} \end{cases}$$ où f et g sont des fonctions de signaux connus et X un signal inconnu.

Un tel système pourra être obtenu à partir de spécifications formelles de type 1 ou de type 2; f et g seront alors des fonctions des signaux d'entrée et X un signal de sortie dont on cherche l'expression formelle.

Il est évident que le système n'admet pas une solution pour toutes les valeurs de f et g; $f = g$, par exemple, conduit à l'absence de solution. Nous nous attacherons donc tout d'abord à déterminer la condition d'existence d'une solution, fonction de f et g. Lorsque cette condition est vérifiée, nous dirons que le système est cohérent. D'un point de vue pratique, dans une approche de synthèse, cette condition nous permettra de détecter des spécifications incohérentes.

Pour un système cohérent, deux cas sont alors possibles :

- Il existe une solution unique et le système sera alors qualifié de complet (sous-entendu "les spécifications dont est issu ce système définissant complètement le signal X ");
- Il existe une infinité de solutions.

Nous établirons dans ce chapitre la condition sur f et g permettant de déterminer si un système est complet ainsi que l'expression de la solution d'un système complet et la forme générale de la solution d'un système incomplètement spécifié.

Pour établir ces deux conditions (de cohérence et de complétude du système) et les deux expressions (solutions d'un système complet et incomplet), nous baserons notre raisonnement sur le fait que notre système à deux relations d'inclusion[1] doit être tel que les propriétés de

[1] $\begin{cases} f \le X \\ g \le \overline{X} \end{cases}$

transitivité et d'anti-symétrie de la relation d'inclusion doivent être vérifiées, si X est solution. Ceci explique la structure de ce chapitre.

Nous allons partir du cas le plus simple jusqu'à la généralisation du problème. Premièrement, nous présentons dans la section suivante le principe de ce raisonnement, à l'aide d'un exemple élémentaire : un système trivial d'équations sur $\mathbb{I}$ qui respecte au départ les trois propriétés ([2.27], [2.28] et [2.29]). Deuxièmement, dans la section 5.3, nous établirons les conditions de cohérence et de complétude du système d'équations. Troisièmement, nous nous intéressons, dans la section 5.4, à la résolution d'un système cohérent mais incomplet. Nous proposerons alors une solution générale et ensuite, une solution particulière nommée "*avec mémoire*". Enfin, la dernière section (5.6) est consacrée à la résolution d'un système dérivé d'un système initial, et qualifié de système étendu avec contrainte de maintien à l'état précédent.

5.2 Utilisation des propriétés de la relation "$\leq$"

Un système d'équations sur $\mathbb{I}$, doit vérifier les trois propriétés de la relation d'ordre partiel : réflexivité ([2.27]),transitivité ([2.28]) et anti-symétrie ([2.29]). La première propriété est toujours vérifiée pour les signaux binaires (voir démonstration de [2.27]). Il ne reste qu'à prouver que les propriétés de transitivité ([2.28]) et d'anti-symétrie ([2.29]) sont vérifiées pour la solution de ce système.

Nous nous intéressons d'abord au système suivant (tel que $g = \overline{f}$) :

$$\begin{cases} 1a) & f \leq X \\ 2a) & \overline{f} \leq \overline{X} \end{cases} \qquad \text{Syst. 1}$$

Ce système trivial admet une solution unique $X = f$ (voir démonstration [2.54] $\Leftrightarrow$ [2.55] à la section 2.3.3). Montrons alors que les propriétés de transitivité et d'anti-symétrie de l'inclusion sont respectées.

- La propriété de transitivité est accomplie si $\begin{cases} f \leq g \\ g \leq h \end{cases} \Rightarrow f \leq h$. En utilisant [2.34] $\Leftrightarrow$ [2.41] nous obtenons : $f \leq X \Leftrightarrow \overline{X} \leq \overline{f}$ et $\overline{f} \leq \overline{X} \Leftrightarrow X \leq f$. En conséquence, il vient que $\forall f, X$ de $\mathbb{I}$:

  Si $f \leq X$ et $X \leq f$, alors $f \leq f$ qui est toujours vrai ([2.27]).

  Si $X \leq f$ et $f \leq X$, alors $X \leq X$ qui est toujours vrai ([2.27]).

  Si $\overline{X} \leq \overline{f}$ et $\overline{f} \leq \overline{X}$, alors $\overline{X} \leq \overline{X}$ qui est toujours vrai ([2.27]).

Si $\overline{f} \le \overline{X}$ et $\overline{X} \le \overline{f}$, alors $\overline{f} \le \overline{f}$ qui est toujours vrai ([2.27]).

donc la propriété de transitivité ([2.28]) est respectée. □

- La propriété d'anti-symétrie est accomplie puisque [2.55] ⇔ [2.54] (voir 2.3.3) :

$$\begin{cases} f \le X \\ \overline{f} \le \overline{X} \end{cases} \Leftrightarrow \quad X = f$$ □

En conclusion, lorsqu'il existe une solution au système 1 étudié, les propriétés de réflexivité ([2.27]), de transitivité ([2.28]) et d'anti-symétrie ([2.29]), sont respectées.

5.3 Conditions de cohérence et de complétude

Soit le système suivant :

$$\begin{cases} 1a) \quad f \le X \\ 2a) \quad g \le \overline{X} \end{cases}$$ Syst. 2

a) Si ce système admet une solution, la propriété de transitivité ([2.28]) doit être vérifiée :

Si $f \le X$ et $X \le \overline{g}$ (en utilisant [2.34] ⇔ [2.41] sur 2a), alors $f \le \overline{g}$, et

Si $g \le \overline{X}$ et $\overline{X} \le \overline{f}$ (en utilisant [2.34] ⇔ [2.41] sur 1a), alors $g \le \overline{f}$.

Pour satisfaire la propriété de transitivité il faut accomplir donc : $\begin{cases} f \le \overline{g} \\ g \le \overline{f} \end{cases}$.

En utilisant [2.34] ⇔ [2.37], [2.1] et [2.10] :

$$\begin{cases} f \le \overline{g} \\ g \le \overline{f} \end{cases} \Leftrightarrow \begin{cases} f \cdot g = 0* \\ g \cdot f = 0* \end{cases} \Leftrightarrow \begin{cases} f \cdot g = 0* \\ f \cdot g = 0* \end{cases} \Leftrightarrow \quad f \cdot g = 0*$$

En conclusion, la propriété de transitivité est accomplie si et seulement si $f \cdot g = 0*$.

b) Nous allons à présent montrer que ce système admet une solution unique si $f = \overline{g}$. La solution unique est alors : $X = f = \overline{g}$.

En effet, si ce système admet une solution, la propriété d'anti-symétrie ([2.29]) doit être vérifiée :

Si nous avons $f \le X$ alors il existe $\overline{f} \le \overline{X}$ tel que $X = f$, et aussi

Si nous avons $g \le \overline{X}$ alors il existe $\overline{g} \le X$ tel que $X = \overline{g}$.

Pour satisfaire la propriété de transitivité (sur l'égalité) il faut accomplir donc : $X = f = \overline{g}$, car si $X = f$ et $X = \overline{g}$ alors $f = \overline{g}$.

Les deux contraintes : la cohérence et la complétude ($f \cdot g = 0^*$ et $f = \overline{g}$, respectivement), peuvent être exprimées avec une seule : $f = \overline{g}$. En effet, ceci est mis en évidence en exprimant $f = \overline{g}$ comme suit :

$$f = \overline{g} \quad \Leftrightarrow \quad \begin{cases} f \cdot g = 0^* \\ \overline{f} \cdot \overline{g} = 0^* \end{cases} \qquad \text{en utilisant [2.54]} \Leftrightarrow \text{[2.57]}$$

En conclusion, pour un système qui n'est définie que par ces deux relations, la propriété d'anti-symétrie est accomplie si et seulement si $f \cdot g = 0^*$ et $\overline{f} \cdot \overline{g} = 0^*$**.**

En définissant la cohérence avec la contrainte $f \cdot g = 0^*$ et la complétude avec la contrainte $\overline{f} \cdot \overline{g} = 0^*$, nous pouvons conclure que :

Ce système a une solution unique, si et seulement si la COHERENCE ($f \cdot g = 0^*$**) et la COMPLETUDE (** $\overline{f} \cdot \overline{g} = 0^*$**) sont toutes les deux respectées (voir le système** 3**).**

En effet, l'existence d'une solution à ce système d'équations est conditionnée, premièrement, par la cohérence laquelle impose que les fonctions f et g, définissant X et $\overline{X}$ (relations 1a et 2a respectivement), soient disjointes : $f \cdot g = 0^*$ et deuxièmement par la complétude qui impose que les fonctions f et g définissent complètement la solution (X) de ce système car $\overline{f} \cdot \overline{g} = 0^*$

$$\begin{cases} \begin{cases} 1a) \quad f \le X \\ 2a) \quad g \le \overline{X} \end{cases} \\ \begin{cases} \textit{cohérence} \quad f \cdot g = 0^* \\ \textit{complétude} \quad \overline{f} \cdot \overline{g} = 0^* \end{cases} \end{cases} \quad \Leftrightarrow \quad X = f = \overline{g} \qquad \text{Syst. 3}$$

5.4 Forme générale de la solution

Nous nous intéressons à présent à un système cohérent ($f \cdot g = 0^*$) mais non complet ($\overline{f} \cdot \overline{g} \neq 0^*$). Nous allons montrer que la solution générale est alors :

$X = f + k_1$ ou bien $X = \overline{g} \cdot \overline{k_2}$.

k_1 et k_2 étant des éléments de $\mathbb{I}$ tels que :

$$\begin{cases} hyp_{1a}) & k_1 \le X \\ hyp_{1b}) & k_2 \le \overline{X} \\ hyp_2) & k_1 \cdot k_2 + f \cdot k_2 + g \cdot k_1 + \overline{f} \cdot \overline{g} \cdot \overline{k_1} \cdot \overline{k_2} = 0* \end{cases} \qquad \text{Syst. 4}$$

Remarque :

Pour tenir compte de l'incomplétude ($\overline{f} \cdot \overline{g} \neq 0*$), nous modifions le système initial, en rajoutant une relation (3a) qui ne modifie pas le système initial car $X + \overline{X} = 1*$.

Il vient alors :

$$\begin{cases} 1a) & f \le X \\ 2a) & g \le \overline{X} \\ 3a) & \overline{f} \cdot \overline{g} \le X + \overline{X} \end{cases} \qquad \text{Syst. 5}$$

DÉMONSTRATION :

Nous allons d'abord introduire hyp_{1a} et hyp_{1b} du système 4 dans le système 5. Nous avons en utilisant [2.46] :

$$\begin{cases} 1a) & f \le X \\ 2a) & g \le \overline{X} \\ 3a) & \overline{f} \cdot \overline{g} \le X + \overline{X} \\ hyp_{1a}) & k_1 \le X \\ hyp_{1b}) & k_2 \le \overline{X} \end{cases} \Leftrightarrow \begin{cases} 1b) & f + k_1 \le X \\ 2b) & g + k_2 \le \overline{X} \\ 3b) & \overline{f} \cdot \overline{g} \le X + \overline{X} \end{cases} \qquad \text{Syst. 6}$$

Pour que le système 6 admette une solution, il faut que ce système soit cohérent :

$$(f + k_1) \cdot (g + k_2) = 0*$$

Cette solution est unique et égale à $X = f + k_1$ ou $X = \overline{g} \cdot \overline{k_2}$ si ce système est complet :

$$\overline{(f + k_1)} \cdot \overline{(g + k_2)} = 0*.$$

D'où :

$$\begin{cases} cohérence) & (f + k_1) \cdot (g + k_2) = 0* \\ complétude) & \overline{(f + k_1)} \cdot \overline{(g + k_2)} = 0* \end{cases}$$

En utilisant [2.3], [2.20], [2.1]; [2.25] nous avons :

$$\begin{cases} (f + k_1) \cdot (g + k_2) = 0* \\ \overline{(f + k_1)} \cdot \overline{(g + k_2)} = 0* \end{cases} \Leftrightarrow \begin{cases} f \cdot g + g \cdot k_1 + f \cdot k_2 + k_1 \cdot k_2 = 0* \\ \overline{f} \cdot \overline{g} \cdot \overline{k_1} \cdot \overline{k_2} = 0* \end{cases}$$

$$\Leftrightarrow \begin{cases} f \cdot g = 0 * \\ k_1 \cdot k_2 + f \cdot k_2 + g \cdot k_1 = 0 * \\ \overline{f} \cdot \overline{g} \cdot \overline{k_1} \cdot \overline{k_2} = 0 * \end{cases}$$

La première égalité représente la contrainte de cohérence du système 5 et la deuxième représente les contraintes de cohérence dues à l'introduction de hyp_{1a} et hyp_{1b} ; la troisième égalité est la contrainte de complétude du système 6. Notons que les deux dernières représentent hyp_2 du système 4.

En résumé, si les contraintes de hyp_2 sont respectées, alors la solution générale d'un système cohérent mais incomplet est :

$X = f + k_1$ ou bien

$X = \overline{g} \cdot \overline{k_2}$

Le dernier résultat est obtenu en utilisant [2.20] : $X = \overline{(g + k_2)} = \overline{g} \cdot \overline{k_2}$

Dans nos travaux, par commodité, nous allons utiliser la première solution décrite par f et k_1 au lieu de la deuxième décrite par $\overline{g}$ et $\overline{k_2}$ comme suit :

$$\begin{cases} \begin{cases} 1a) & f \le X \\ 2a) & g \le \overline{X} \\ 3a) & \overline{f} \cdot \overline{g} \le X + \overline{X} \end{cases} \\ \{cohérence) \quad f \cdot g = 0 * \quad \Leftrightarrow \quad X = f + k_1 \\ \begin{cases} hyp_{1a}) & k_1 \le X \\ hyp_{1b}) & k_2 \le \overline{X} \end{cases} \\ hyp_2) \quad k_1 \cdot k_2 + f \cdot k_2 + g \cdot k_1 + \overline{f} \cdot \overline{g} \cdot \overline{k_1} \cdot \overline{k_2} = 0 * \end{cases} \qquad \text{Syst. 7}$$

Il est possible d'avoir une infinité de solutions qui respectent ces contraintes. Nous présentons succinctement 4 exemples.

EXEMPLES :

k_1 et k_2 peuvent être tels que $k_1 + k_2 = \overline{f} \cdot \overline{g}$, par exemple :

1) $k_1 = 0 *$, $k_2 = \overline{f} \cdot \overline{g}$ ou bien

2) $k_1 = \overline{f} \cdot \overline{g} \cdot \overline{h}$, $k_2 = \overline{f} \cdot \overline{g} \cdot h$ (h élément quelconque de $\mathbb{I}$)

Cependant, cette contrainte n'est pas obligatoire, par exemple :

3) $k_1 = f \cdot \overline{g}$, $k_2 = \overline{f} \cdot \overline{g}$ où $k_1 + k_2 = \overline{g}$

4) $k_1 = \overline{f} \cdot \overline{g} \cdot \overline{h}$, $k_2 = g + \overline{f} \cdot h$ où $k_1 + k_2 = \overline{f}$

Il faut noter que les quatre exemples présentés accomplissent hyp_2, en supposant que la cohérence ($f \cdot g = 0*$) est respectée :

1) $(0*) \cdot (\overline{f} \cdot \overline{g}) + f \cdot (\overline{f} \cdot \overline{g}) + g \cdot (0*) + \overline{f} \cdot \overline{g} \cdot \overline{(0*)} \cdot \overline{\left(\overline{f} \cdot \overline{g}\right)} = 0*$

2) $(\overline{f} \cdot \overline{g} \cdot \overline{h}) \cdot (\overline{f} \cdot \overline{g} \cdot h) + f \cdot (\overline{f} \cdot \overline{g} \cdot h) + g \cdot (\overline{f} \cdot \overline{g} \cdot \overline{h}) + \overline{f} \cdot \overline{g} \cdot \overline{\left(\overline{f} \cdot \overline{g} \cdot \overline{h}\right)} \cdot \overline{\left(\overline{f} \cdot \overline{g} \cdot h\right)} = 0*$

3) $(f \cdot \overline{g}) \cdot (\overline{f} \cdot \overline{g}) + f \cdot (\overline{f} \cdot \overline{g}) + g \cdot (f \cdot \overline{g}) + \overline{f} \cdot \overline{g} \cdot \overline{\left(f \cdot \overline{g}\right)} \cdot \overline{\left(\overline{f} \cdot \overline{g}\right)} = 0*$

4) $(\overline{f} \cdot \overline{g} \cdot \overline{h}) \cdot (g + \overline{f} \cdot h) + f \cdot (g + \overline{f} \cdot h) + g \cdot (\overline{f} \cdot \overline{g} \cdot \overline{h}) + \overline{f} \cdot \overline{g} \cdot \overline{\left(\overline{f} \cdot \overline{g} \cdot \overline{h}\right)} \cdot \overline{\left(g + \overline{f} \cdot h\right)} = 0*$

Après simplification, d'après le système 7 les solutions de ces exemples sont donc respectivement :

1) $X = f + 0* = f$

2) $X = f + \overline{f} \cdot \overline{g} \cdot \overline{h} = f \cdot \overline{g} + \overline{f} \cdot \overline{g} \cdot \overline{h} = \overline{g} \cdot (f + \overline{f} \cdot \overline{h}) = \overline{g} \cdot (f + \overline{h})$

3) $X = f + f \cdot \overline{g} = f$

4) $X = f + \overline{f} \cdot \overline{g} \cdot \overline{h} = f \cdot \overline{g} + \overline{f} \cdot \overline{g} \cdot \overline{h} = \overline{g} \cdot (f + \overline{f} \cdot \overline{h}) = \overline{g} \cdot (f + \overline{h})$

Nous allons de plus montrer dans la section suivante que les signaux $RS(f,g)$ et $SR(f,g)$ sont des solutions particulières.

5.5 Utilisation des opérations SR et RS pour l'expression de solutions particulières

5.5.1 Introduction

A partir de la solution générale proposée (système 7) nous constatons qu'il est possible de proposer des solutions à un système cohérent mais incomplet, en utilisant des opérations temporelles pour définir k_1 et k_2, comme par exemple : le front montant, le front descendant, le TON et le TOF.

Nous allons montrer dans cette section l'existence d'une solution temporelle particulière que nous appelons *avec mémoire*. Cette solution exprimée en termes de variables logiques est :

$$\forall t \in \mathbb{R}^{+*}, \quad X(t) = f(t) \vee \left[\exists t_1 < t : \left(\left(f(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\overline{g})(d) = 1 \right) \right) \right]$$

et peut s'écrire à l'aide des opérations SR et RS : $X = SR(f, g) = RS(f, g)$.

5.5.2 Résolution

Nous allons définir d'abord les signaux k_3 et k_4 par des prédicats en fonction des valeurs des signaux f et g à l'instant t et de l'existence, dans le passé, des combinaisons possibles de ces signaux. Les signaux k_3 et k_4 sont définis de la façon suivante (voir figure 30) :

$$\forall t \in \mathbb{R}^{+*},$$

$$k_3(t) = \left[\exists t_1 < t : \left(\left(f(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\overline{g})(d) = 1 \right) \right) \right]$$

$$k_{4a}(t) = \left[\exists t_1 < t : \left(\left(g(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\overline{f})(d) = 1 \right) \right) \right]$$

$$k_{4b}(t) = \left[\forall d \in]0, t], (\overline{f} \cdot \overline{g})(d) = 1 \right]$$

Où, $k_4 = k_{4a} + k_{4b}$.

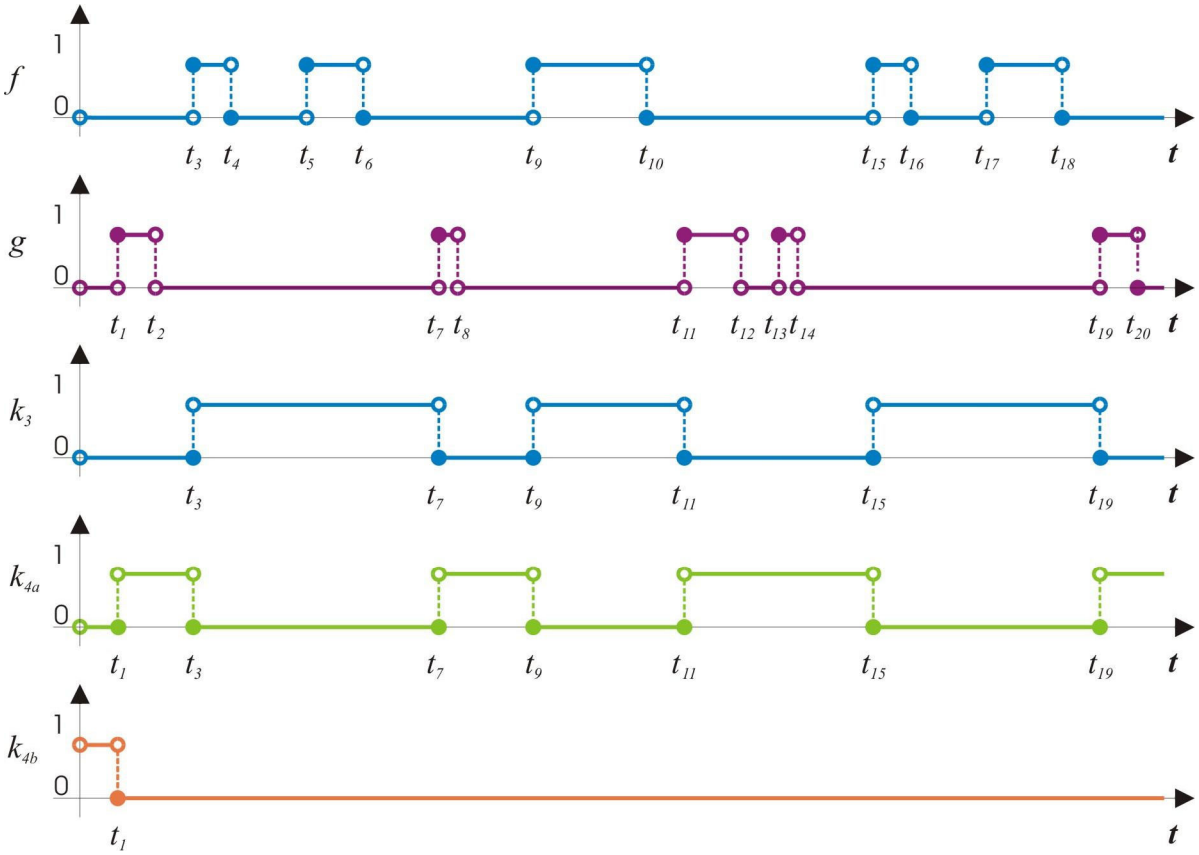


Figure 30 *(a) Signal f (b) Signal g (c) Signal k3 (d) Signal k4a (e) Signal k4b*

Le prédicat k_3 est égal à 1 (voir figure 30c), durant les périodes $]t_1,t]$ telles que $f(t_1)=1$ et $\neg g(t)=1$. Le prédicat k_{4a} est égal à 1 (voir figure 30d), durant les périodes $]t_1,t]$, telles que $\neg g(t_1)=1$ et $\neg f(t)=1$. Le prédicat k_{4b} est égal à 1 (voir figure 30e), dès que $t>0$ et pendant que $(\overline{f}\cdot\overline{g})(d)=1$.

On peut démontrer que ces prédicats sont toujours définis et qu'ils vérifient hyp_2, si $f\cdot g=0*$ (condition de cohérence) et $\overline{f}\cdot\overline{g}\neq 0*$ (système incomplet). La démonstration est dans l'annexe D).

Ainsi, pour un système incomplet tel que 5, après vérification de la cohérence ($f\cdot g=0*$), il est toujours possible de proposer la solution particulière $X=f+k_3$, nommée *avec mémoire* (voir système 7) *:*

$$\left\{\begin{array}{ll}\left\{\begin{array}{ll}1a) & f\le X\\ 2a) & g\le\overline{X}\\ 3a) & \overline{f}\cdot\overline{g}\le X+\overline{X}\end{array}\right.\\ \left\{cohérence\right) & f\cdot g=0* \quad\Leftrightarrow\quad X=f+k_3\\ \left\{\begin{array}{ll}hyp_{1a}) & k_3\le X\\ hyp_{1b}) & k_4\le\overline{X}\end{array}\right.\\ hyp_2) & k_3\cdot k_4+f\cdot k_4+g\cdot k_3+\overline{f}\cdot\overline{g}\cdot\overline{k_3}\cdot\overline{k_4}=0* \quad\text{(toujours vraie)}\end{array}\right. \qquad \text{Syst. 8}$$

Où la solution $X=f+k_3$ est définie en variables logiques comme suit :

$$\forall t\in\mathbb{R}^{+*},\quad X(t)=f(t)\vee\left[\exists t_1<t:\left(\left(f(t_1)=1\right)\wedge\left(\forall d\in]t_1,t],(\overline{g})(d)=1\right)\right)\right]$$

Remarque :

Etant donné que hyp_2 est toujours vraie, nous en déduisons que ce système reste cohérent $(f+k_3)\cdot(g+k_4)=0*$ car $f\cdot g+k_3\cdot k_4+f\cdot k_4+g\cdot k_3=0*$ et qu'il a été complété par les paramètres k_3 et k_4 choisis, car $\overline{f}\cdot\overline{g}\cdot\overline{k_3}\cdot\overline{k_4}=0*$.

5.5.3 L'opération REP et sa relation avec la solution avec mémoire

5.5.3.1 Définition de l'opération REP

Nous allons définir dans cette section une nouvelle opération nommée *REP* qui signifie "rester dans l'état précédent". Cette opération nous servira à simplifier l'expression de la solution *avec mémoire*. Cette opération est définie comme suit :

Définition 13 : Opération *REP*

$$\begin{aligned} \mathbb{I}^2 &\rightarrow \mathbb{I} \\ (s,r) &\mapsto REP(s,r) \end{aligned}$$

Où $\forall t \in \mathbb{R}^{+*}$, $REP(s,r)(t) = \left(\begin{array}{l} \left[\exists t_1 < t : \left(\left(s(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\bar{r})(d) = 1 \right) \right) \right] \\ \quad \vee \left[\exists t_1 < t : \left(\left(r(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\bar{s})(d) = 1 \right) \right) \right] \\ \quad \vee \left[\forall d \in]0, t], (\bar{s} \cdot \bar{r})(d) = 1 \right] \end{array} \right)$

La figure suivante (figure 31) nous montre les signaux f, g, les résultats des opérations $REP(f,g)$ et $SR(f,g)$. Nous notons que c'est le paramètre k_{4b} (voir figure 30) qui définit la période initiale où $SR(f,g)$ vaut 0 et en plus $SR(f,g)$ **maintient la valeur précédente** quand $REP(f,g)$ vaut 1. Ce sont les paramètres k_3 et k_4 qui définissent la valeur de $SR(f,g)$ à travers les relations hyp_{1a} et hyp_{1b}.

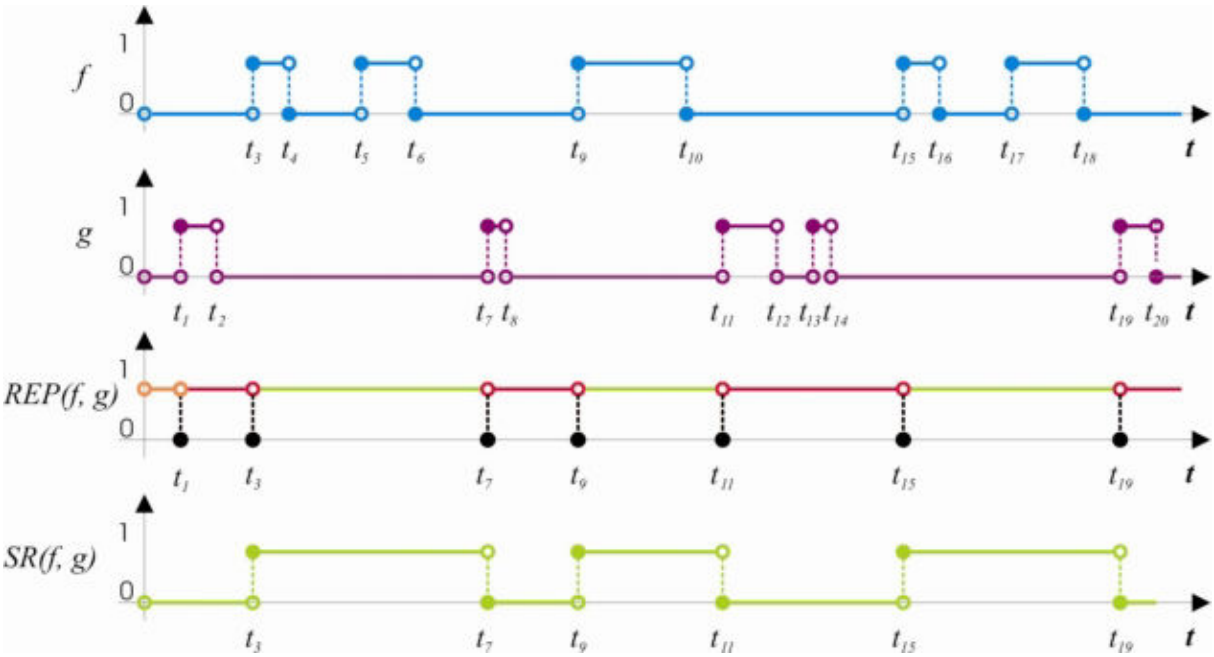


Figure 31 *Représentation graphique de l'opération REP(f,g) et SR(f,g)*

5.5.3.2 Spécification SF-T5

L'opération REP nous permet d'introduire ci-dessous une nouvelle classe de spécification qui s'ajoute à celles présentées au chapitre 4. L'opération REP est définie à partir de deux signaux (f et g) et contraint un troisième signal (X) à rester dans l'état tel qu'il est montré dans la

section 5.5.3.4. Afin d'alléger la notation nous noterons REP_X pour $REP(f,g)$ dans un système définissant X.

Classification	**Représentation générale formelle en $\mathbb{I}$**
SF-T5	$F_1(u) \le REP_{X_i}$ Où : $F_1(u)$ est une fonction d'entrées

Cette spécification est une relation entre une combinaison d'entrées et le résultat de l'opération REP qui contraint le signal X_i à rester dans l'état.

5.5.3.3 Théorèmes relatifs aux opérations SR et RS

En utilisant l'opération REP, il est possible d'établir les théorèmes suivants ([5.1] à [5.4]), relatifs aux opérations SR et RS (voir démonstrations à l'annexe D). Ces quatre théorèmes montrent les systèmes d'équations qui ont comme solution les opérations SR et RS.

$$\begin{cases} \begin{cases} 1a) & s \le X \\ 2a) & r \le \overline{X} \\ 3a) & m \le REP_X \end{cases} \\ \begin{cases} 1a) & k_s \le X \\ 2a) & k_r \le \overline{X} \\ 3a) & k_m \le REP_X \end{cases} \\ \begin{cases} hyp_1) & s \cdot r + s \cdot m + r \cdot m = 0* \\ hyp_2) & k_s \cdot k_r + k_s \cdot k_m + k_r \cdot k_m = 0* \\ hyp_3) & \left(\begin{array}{r} s \cdot (k_r + k_m) + r \cdot (k_s + k_m) \\ + m \cdot (k_s + k_r) \end{array} \right) = 0* \\ hyp_4) & \bar{s} \cdot \bar{r} \cdot \bar{m} \cdot \overline{k_s} \cdot \overline{k_r} \cdot \overline{k_m} = 0* \end{cases} \end{cases} \Leftrightarrow \begin{array}{c} X = SR(s + k_s, r + k_r) \\ \text{ou bien} \\ X = RS(s + k_s, r + k_r) \end{array} \qquad [5.1]$$

$$\begin{cases} 1a) & s \le X \\ 2a) & r \le \overline{X} \\ 3a) & \bar{s} \cdot \bar{r} \le REP_X \\ coh\acute{e}rence) & s \cdot r = 0* \end{cases} \Leftrightarrow \begin{array}{c} X = SR(s,r) \\ \text{ou bien} \\ X = RS(s,r) \end{array} \qquad [5.2]$$

$$\begin{cases} 1a) & s \le X \\ 2a) & \bar{s} \cdot r \le \overline{X} \\ 3a) & \bar{s} \cdot \bar{r} \le REP_X \end{cases} \Leftrightarrow X = SR(s,r) \qquad [5.3]$$

$$\begin{cases} 1a) & s \cdot \overline{r} \le X \\ 2a) & r \le \overline{X} \\ 3a) & \overline{s} \cdot \overline{r} \le REP_X \end{cases} \quad \Leftrightarrow \quad X = RS(s,r) \qquad [5.4]$$

5.5.3.4 Solution avec mémoire

Nous pouvons simplifier la notation de la solution avec mémoire, parce que $REP(f,g) = k_3 + k_4$:

$$\forall t \in \mathbb{R}^{+*},\ REP(f,g)(t) = \left(\begin{array}{l} \left[\exists t_1 < t : \left(\left(f(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\overline{g})(d) = 1 \right) \right) \right] \\ \quad \vee \left[\exists t_1 < t : \left(\left(g(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\overline{f})(d) = 1 \right) \right) \right] \\ \quad \vee \left[\forall d \in]0, t], (\overline{f} \cdot \overline{g})(d) = 1 \right] \end{array} \right)$$

Nous pouvons simplifier le système 8, en sachant que l'hypothèse 2 de la solution *avec mémoire* est toujours vraie. En effet, d'après le système 7 :

$$\begin{cases} 3a) & \overline{f} \cdot \overline{g} \le X + \overline{X} \\ hyp_{2d}) & \overline{f} \cdot \overline{g} \cdot \overline{k_3} \cdot \overline{k_4} = 0* \end{cases} \Leftrightarrow \begin{cases} \overline{f} \cdot \overline{g} \le X + \overline{X} \\ \overline{f} \cdot \overline{g} \le k_3 + k_4 \end{cases}$$

$$\Leftrightarrow \quad \left\{ \overline{f} \cdot \overline{g} \le k_3 + k_4 \right.$$

$$\Leftrightarrow \quad \left\{ \overline{f} \cdot \overline{g} \le REP(f,g) \right.$$

Par conséquent, le système 8 peut être réécrit, en utilisant le nouvel type de spécification SF-T5, comme suit :

$$\left\{ \begin{array}{l} \left\{ \begin{array}{ll} 1a) & f \le X \\ 2a) & g \le \overline{X} \\ 3a \cdot hyp_{2d}) & \overline{f} \cdot \overline{g} \le REP_X \end{array} \right. \\ \left\{ cohérence) \quad f \cdot g = 0* \quad \Leftrightarrow \quad X = f + k_3 \right. \\ \left\{ \begin{array}{ll} hyp_{1a}) & k_3 \le X \\ hyp_{1b}) & k_4 \le \overline{X} \\ hyp_2) & k_3 \cdot k_4 + f \cdot k_4 + g \cdot k_3 + \overline{f} \cdot \overline{g} \cdot \overline{k_3} \cdot \overline{k_4} = 0* \quad \text{(toujours vraie)} \end{array} \right. \end{array} \right. \qquad \text{Syst. 9}$$

REMARQUES :

1) Le système 9, a une solution où le choix ($\overline{f} \cdot \overline{g}$) demande de rester dans l'état précédent : $REP(f,g)$, **avec une condition initiale définie par** k_{4b}. Autrement dit :

- Dès l'instant où f prend la valeur 1, $X = 1$ (point 1a), le signal X restera dans cet état **jusqu'à** l'instant où g prend la valeur 1.
- Dès l'instant où g prend la valeur 1, $X = 0$ (point 1b), le signal X restera dans cet état **jusqu'à** l'instant où f prend la valeur 1.
- **D'autre part, pendant la partie initiale où ni f ni g ne prennent encore la valeur 1 alors la condition initiale est établie pour prendre la valeur $X = 0$, grâce au choix du k_{4b} (hyp_2).**

En conclusion, nous pouvons simplifier la notation du système 9 d'après [5.2] :

$$\begin{cases} 1a) & f \le X \\ 2a) & g \le \overline{X} \\ 3a) & \overline{f} \cdot \overline{g} \le REP_X \\ coh\acute{e}rence) & f \cdot g = 0 * \end{cases} \quad \Leftrightarrow \quad X = SR(f,g) = RS(f,g)$$

5.6 Système étendu avec contrainte de maintien dans l'état précédent

Nous allons nous intéresser dans cette section à la résolution du système ci-dessous :

$$\begin{cases} 1a) & f \le X \\ 2a) & g \le \overline{X} \\ 3a) & h \le REP_X \end{cases} \qquad \text{Syst. 10}$$

où f, g et h sont des signaux connus, X un signal inconnu et k_f, k_g et k_h des paramètres.

Ce système peut être utile lorsqu'une SF impose qu'un signal reste dans l'état précédent comme nous le verrons dans les chapitres suivants (inclusion 3a).

Pour trouver les conditions de cohérence et complétude de ce système, nous introduisons des contraintes sur les paramètres qui conduisent au système suivant :

$$\begin{cases} 1a) & f \le X \\ 2a) & g \le \overline{X} \\ 3a) & h \le REP_X \\ 4a) & k_f \le X \\ 5a) & k_g \le \overline{X} \\ 6a) & k_h \le REP_X \end{cases} \quad \Leftrightarrow \quad \begin{cases} 1a \cdot 4a) & f + k_f \le X \\ 2a \cdot 5a) & g + k_g \le \overline{X} \\ 3a \cdot 6a) & h + k_h \le REP_X \end{cases}$$

D'après [5.2], la solution de ce système est :

$$X = SR\Big((f + k_f),(g + k_g)\Big) = RS\Big((f + k_f),(g + k_g)\Big)$$

Si et seulement si :

- Le système est cohérent : $(f + k_f)\cdot(g + k_g) = 0^*$
- Les signaux connus f, g et h sont tels que : $\overline{(f + k_f)}\cdot\overline{(g + k_g)} = (h + k_h)$

En développant la première expression et en utilisant [2.54] $\Leftrightarrow$ [2.58] pour la deuxième, nous avons

$$\begin{cases} f\cdot g + f\cdot k_g + g\cdot k_f + k_f\cdot k_g = 0^* \\ \overline{(f + k_f)}\cdot\overline{(g + k_g)}\cdot\overline{(h + k_h)} + \Big((f + k_f) + (g + k_g)\Big)\cdot(h + k_h) = 0^* \end{cases}$$

En développant ces expressions et en utilisant [2.25], nous avons

$$\begin{cases} hyp_1) & f\cdot g + f\cdot h + g\cdot h = 0^* \\ hyp_2) & k_f\cdot k_g + k_f\cdot k_h + k_g\cdot k_h = 0^* \\ hyp_3) & f\cdot(k_g + k_h) + g\cdot(k_f + k_h) + h\cdot(k_f + k_g) = 0^* \\ hyp_4) & \overline{f}\cdot\overline{g}\cdot\overline{h}\cdot\overline{k_f}\cdot\overline{k_g}\cdot\overline{k_h} = 0^* \end{cases}$$

En conclusion, si et seulement si les hypothèses 1 à 4 sont respectées alors il existe une solution avec mémoire :

$$\begin{cases} \begin{cases} 1a) & f \le X \\ 2a) & g \le \overline{X} \\ 3a) & h \le REP_X \\ 4a) & \overline{f}\cdot\overline{g}\cdot\overline{h} \ne 0^* \end{cases} \\ \begin{cases} 5a) & k_f \le X \\ 6a) & k_g \le \overline{X} \\ 7a) & k_h \le REP_X \end{cases} \\ \begin{cases} hyp_1) & f\cdot g + f\cdot h + g\cdot h = 0^* \\ hyp_2) & k_f\cdot k_g + k_f\cdot k_h + k_g\cdot k_h = 0^* \\ hyp_3) & f\cdot(k_g + k_h) + g\cdot(k_f + k_h) + h\cdot(k_f + k_g) = 0^* \\ hyp_4) & \overline{f}\cdot\overline{g}\cdot\overline{h}\cdot\overline{k_f}\cdot\overline{k_g}\cdot\overline{k_h} = 0^* \end{cases} \end{cases} \Leftrightarrow \begin{cases} X = SR\Big((f + k_f),(g + k_g)\Big) \\ \text{ou bien} \\ X = RS\Big((f + k_f),(g + k_g)\Big) \end{cases} \quad \text{Syst. 11}$$

Remarques :

Il faut noter que la solution trouvée ici est identique à celle de la section 5.5 ($k_f = k_g = k_h = 0^*$). En effet, si $k_f = k_g = k_h = 0^*$ alors

$$\begin{cases} hyp_1) & f \cdot g + f \cdot h + g \cdot h = 0* \\ hyp_2) & 0* = 0* \\ hyp_3) & 0* = 0* \\ hyp_4) & \overline{f} \cdot \overline{g} \cdot \overline{h} = 0* \end{cases} \Leftrightarrow \begin{cases} hyp_{1a}) & f \cdot g = 0* \\ hyp_{1b}) & h \cdot (f + g) = 0* \\ hyp_4) & \overline{h} \cdot (\overline{f} \cdot \overline{g}) = 0* \end{cases} \Leftrightarrow \begin{cases} f \cdot g = 0* \\ h = \overline{f} \cdot \overline{g} \end{cases}$$

5.7 Conclusion

Nous avons démontré que le système étudié admet une solution si $f \cdot g = 0*$ (condition de cohérence). Pour un système cohérent, nous avons également déterminé une condition de complétude ($\overline{f} \cdot \overline{g} = 0*$).

Lorsque le système est cohérent mais incomplet, nous avons proposé une forme générale de la solution ainsi qu'une forme particulière qui est basée sur les opérations mémoire SR et RS.

Dans le prochain chapitre nous allons proposer une méthode de synthèse pour concevoir la commande d'un système logique séquentiel en utilisant ces résultats théoriques.

Chapitre 6

Nous présentons dans ce chapitre la méthode de synthèse de fonctions de contrôle à partir de spécifications formelles jusqu'à l'implantation de la commande. Cette méthode s'appuie sur la vérification algébrique des propriétés du système formel obtenu à partir des spécifications. Nous détaillerons en plus trois algorithmes de vérification/correction des propriétés afin de calculer la commande.

6. Synthèse d'un contrôleur logique à partir des spécifications formelles

Nous allons présenter dans ce chapitre la deuxième partie de la méthode d'élaboration de la commande. Cette partie (voir la figure 32) concerne la conception formelle de la commande à partir de la SF sur $\mathbb{I}$. L'objectif est de générer automatiquement les fonctions de contrôle sûres implantables sur le calculateur industriel désiré qui les exécute en parallèle ou séquentiellement.

De plus, nous considérerons une communication avec l'utilisateur afin de surmonter le non accomplissement des propriétés telles que la cohérence et la complétude. En effet, il est envisagé que la SF peut contenir des erreurs ou des oublis car pour des gros systèmes il serait impensable de supposer que l'utilisateur puisse concevoir une SF sans équivoque et complète.

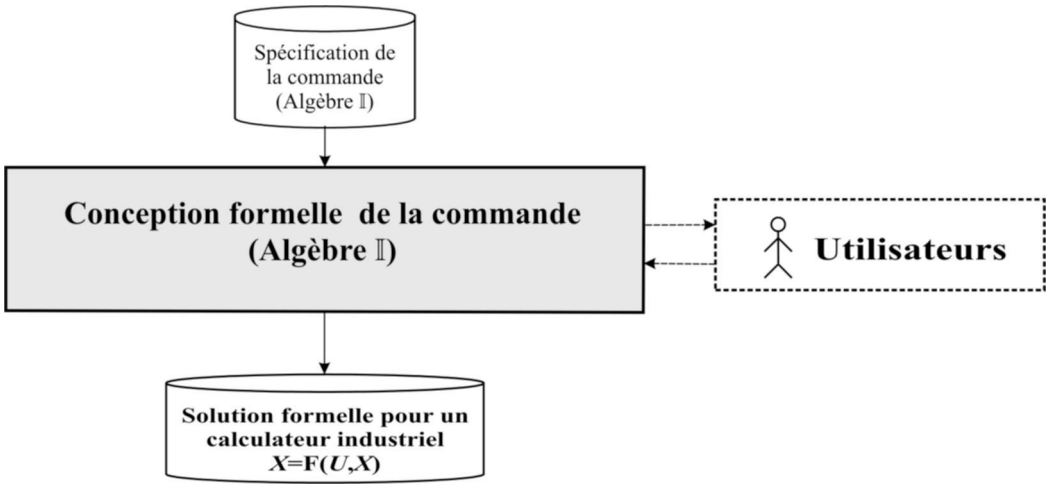


Figure 32 *Objectif du chapitre : Conception formelle de la commande en algèbre $\mathbb{I}$*

La méthode proposée, décrite au chapitre 5, repose sur la résolution d'un système d'équations, écrites en algèbre $\mathbb{I}$. Cependant les spécifications formelles peuvent concerner plusieurs sorties du contrôleur, le système global à résoudre représente les contraintes imposées par la SF où les sorties du système de contrôle-commande constituent les inconnues du système. L'ensemble de ces équations représente donc le comportement attendu du SED étudié. Chacune d'entre-elles correspond à un élément de la spécification de fonctionnement. La fonction de contrôle recherchée est donc la solution du système d'équations représentant la SF.

Dans ce chapitre, nous avons choisi d'introduire d'abord les quatre principaux pas qui décrivent la méthode. Les propriétés, que nous allons vérifier, sont présentées ensuite en 6.2 en incluant les expressions qui expriment les contraintes pour vérifier ces propriétés et les actions à suivre pour faire la correction. Dans la section qui suit (6.3) la méthode est détaillée. La démarche de vérification/correction prévoit l'utilisation de 3 algorithmes généraux de vérification et correction des propriétés qui sont présentés en 6.4 en utilisant des diagrammes de flux.

6.1 Introduction à la méthode

Les quatre pas principaux qui décrivent la méthode sont présentés dans la figure suivante.

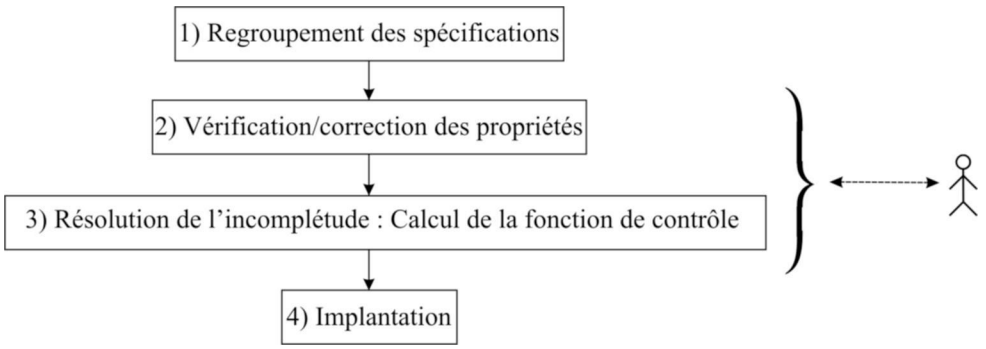


Figure 33 *Présentation générale de la méthode de conception formelle de la commande*

1) REGROUPEMENT DES SPÉCIFICATIONS.

Les spécifications sont regroupées de façon à obtenir :

Un système des sorties qui regroupe toutes relations de la SF qui contraint chaque sortie X_i du contrôleur un **système des sorties** tel que :

$$\text{A)} \begin{cases} 1a) & f_{xi} \le X_i \\ 2a) & g_{xi} \le \overline{X_i} \\ 3a) & h_{xi} \le REP_{Xi} \end{cases}$$

Un système du comportement global tel que le précédent mais concernant de fonctions de sorties $F(X)$. Ce système regroupe toutes les relations de la SF qui contraient le comportement du système tel que :

$$\text{B)} \begin{cases} 1b) & f_{F1(X)} \le F_1(X) \\ 2b) & g_{F1(X)} \le \overline{F_1(X)} \\ 3b) & h_{F1(X)} \le REP_{F1(X)} \end{cases}$$

Il y a en plus deux relations d'égalité qui regroupent d'une part **les hypothèses sur les**

entrées (SF-T3) et d'autre part **les états interdits** (SF-T4) :

$$\begin{cases} 4) & U_O = 0^* \\ 5) & X_O = 0^* \end{cases} \qquad \text{où } U_O = \sum F(u) \text{ des SF-T3 et } X_O = \sum F(X) \text{ des SF-T4.}$$

2) **VÉRIFICATION ET CORRECTION DES PROPRIÉTÉS**. On analyse la SF, en vérifiant le respect des propriétés. A chaque propriété non vérifiée nous identifions les causes du non-respect. En conséquence, des corrections peuvent être proposées par l'utilisateur en ajoutant ou en enlevant des assertions.

3) **LE CALCUL DE LA FONCTION DE CONTRÔLE**. Dans cette étape nous complétons d'abord les systèmes corrigés précédemment avec des équations faisant intervenir les paramètres k_f, k_g et k_h pour chaque système de sortie X_i : k_{fxi}, k_{gxi} et k_{hxi}. Comme le choix dans l'incomplétude est limité par la vérification des propriétés à respecter et en plus par le mode d'exécution du contrôleur, nous calculerons alors dans un premier temps les contraintes à vérifier par ces paramètres et ensuite nous proposons une démarche pour la recherche d'une solution.

4) **L'IMPLANTATION DE LA FONCTION DE CONTRÔLE**. Finalement, nous considérons l'implantation de la fonction de contrôle pour un calculateur exécutant les fonctions de contrôle en parallèle ou séquentiellement.

Avant de détailler la méthode, nous allons présenter dans la section suivante les propriétés à respecter et les expressions à utiliser pour vérifier l'accomplissement de ces propriétés individuellement.

6.2 Propriétés

La méthode proposée repose aussi sur le respect de plusieurs propriétés. Nous avons deux types de propriétés :

a) Les propriétés intrinsèques qui sont des propriétés inhérentes au problème à résoudre et nous n'avons considéré que la vérification des états interdits par la SF.

b) Les propriétés extrinsèques qui sont des propriétés qui doivent être respectées pour tous les systèmes d'équations. Nous avons inclus dans nos travaux 5 propriétés de ce type : la *cohérence*, la *complétude,* le *non blocage, l'unité stabilité* et la *semi-insensibilité à l'ordre.* La *cohérence* et la *complétude* sont des propriétés indispensables pour avoir une solution unique d'un système d'équations (voir la section 5.3). Le *non blocage* des fonctions de sortie, est une propriété qu'il faudrait respecter pour avoir une solution correctement spécifiée. Les deux

dernières : l'unité stabilité et la semi-insensibilité à l'ordre, sont des propriétés demandées pour l'implantation sur un calculateur qui exécute les fonctions de contrôle séquentiellement[1] [Med 00], [Yed 00a].

L'ensemble d'équations doit alors respecter ces propriétés afin que la fonction solution de chaque sortie (systèmes de la forme A) soit implantable et correctement spécifiée. Or, il suffit de bien traduire ces propriétés en algèbre $\mathbb{I}$ afin de garantir par avance le respect de ces propriétés. Ces propriétés sont exprimées par 1* (toujours) ou bien par 0* (jamais) car [2.54] $\Leftrightarrow$ [2.56].

Il existe deux moyens de faire les corrections, afin de surmonter le non accomplissement des propriétés soit en enlevant des assertions au système soit en ajoutant des assertions.

6.2.1 Propriétés extrinsèques

Nous allons donner d'abord les concepts de ces propriétés et ensuite nous donnerons les équations qui définissent le respect de chaque propriété en indiquant le moyen de les corriger. D'abord, nous définirons les expressions qui serviront à vérifier les propriétés de la SF à partir des systèmes du type A (6.1). Ensuite, nous définirons les expressions qui serviront à vérifier les propriétés du système complété avec des équations faisant intervenir des paramètres k_{fxi}, k_{gxi} et k_{hxi} de chaque système de sortie comme suit :

$$\text{C)} \quad \begin{cases} F_{xi} \le X_i \\ G_{xi} \le \overline{X_i} \\ H_{xi} \le REP_{Xi} \end{cases} \qquad où \qquad \begin{aligned} F_{xi} &= f_{xi} + k_{fxi} \\ G_{xi} &= g_{xi} + k_{gxi} \\ H_{xi} &= h_{xi} + k_{hxi} \end{aligned}$$

Nous noterons F, G et H les variables du système complété

Rappelons que l'*ajout d'assertions* afin de compléter le système a été présenté dans le chapitre précédent.

COHERENCE

Un système d'équations bien spécifié doit être cohérent pour avoir une solution. Lorsqu'aucun signal binaire de sortie ne peut être solution d'un système d'équations sans l'ajout d'hypothèses restrictives sur le comportement des signaux d'entrée, la spécification proposée correspond à un problème sur-contraint, dû à la présence d'éléments de spécifications partiellement incohérents entre eux. D'après [5.1], nous avons

[1] Rappelons que la semi-confluence n'est pas nécessaire pour l'exécution dans un seul calculateur.

☞ • Un système A est cohérent si $hyp_1 = 0*$ où $hyp_1 = f_{xi} \cdot g_{xi} + f_{xi} \cdot h_{xi} + g_{xi} \cdot h_{xi}$.

• Il peut être complété si le système (système C) reste cohérent : $hyp_2 + hyp_3 = 0*$ où

$$\begin{cases} hyp_2 = k_{fxi} \cdot k_{gxi} + k_{fxi} \cdot k_{hxi} + k_{gxi} \cdot k_{hxi} \\ hyp_3 = f_{xi} \cdot (k_{gxi} + k_{hxi}) + g_{xi} \cdot (k_{fxi} + k_{hxi}) + h_{xi} \cdot (k_{fxi} + k_{gxi}) \end{cases}$$

Si le système est incohérent *il faut enlever ou modifier des assertions* afin de rendre le système cohérent. Nous proposons l'algorithme 1 pour la vérification/correction de cette propriété (en 6.4.1).

COMPLETUDE

Un système d'équations bien spécifié doit être complet pour que ce système ait une solution unique qui puisse être implantée. En effet, la définition d'une fonction solution parmi une famille de solutions implique un choix dans l'incomplétude. D'après [5.1], nous avons

☞ • Un système A est complet si $hyp_4 = 0*$ où $hyp_4 = \overline{f_{xi}} \cdot \overline{g_{xi}} \cdot \overline{h_{xi}}$.

• Pour le système complété (système C), il faut que

$hyp_4 = 0*$ où $hyp_4 = \overline{f_{xi}} \cdot \overline{g_{xi}} \cdot \overline{h_{xi}} \cdot \overline{k_{fxi}} \cdot \overline{k_{gxi}} \cdot \overline{k_{hxi}}$

Nous proposons la démarche décrite en 6.3.3 pour la vérification/correction de cette propriété.

NON BLOCAGE

Le non blocage est une propriété des fonctions de contrôle de sorties qui doit être respectée. Un signal de sortie est bloquant s'il peut rester toujours vrai ou bien toujours faux. Par exemple, les signaux $X = SR(f, 0*)$ et $X = SR(0*, g)$ sont bloquants.

Pour garantir le non blocage de la commande, il suffit de prouver qu'il existe des transitions définissant la mise à zéro et la mise à un de chaque sortie. En effet,

☞ • Pour vérifier pendant la synthèse qu'une sortie X_i est non bloquante, nous devons prouver que le système d'assertions, qui définit ce signal, comporte à la fois :

a) au moins une assertion $f_{xi} \le X_i$ tel que $f_{xi} \cdot \overline{U_0} \neq 0*$ et

b) au moins une assertion $g_{xi} \le \overline{X_i}$ tel que $g_{xi} \cdot \overline{U_0} \neq 0*$

• Le système complété (système C) vérifie le non blocage si :

a) $k_{fxi} \cdot \overline{U_0} \neq 0*$, si $f_{xi} \cdot \overline{U_0} = 0*$ et

b) $k_{gxi} \cdot \overline{U_0} \neq 0*$, si $g_{xi} \cdot \overline{U_0} = 0*$

Si la fonction est bloquante *il faut ajouter des assertions* afin de garantir le respect de cette propriété. Nous proposons l'algorithme 3 pour la vérification/correction de cette propriété (en 6.4.3).

SEMI-INSENSIBILITE A L'ORDRE

La semi-insensibilité à l'ordre est une propriété qui n'est demandée qu'aux fonctions à implanter sur un calculateur qui exécute ces fonctions séquentiellement.
Les systèmes où les fonctions de sorties sont toutes indépendantes entre elles sont insensibles à l'ordre et en conséquence semi-insensibles à l'ordre. Cependant, la recherche d'au moins un ordre de calcul qui respecte cette propriété pour des systèmes où les sorties sont dépendantes n'est pas évident.

Nous ne traiterons dans ce travail que les fonctions de sorties dont l'ordonnancement peut être déduit à partir d'un graphe de dépendances, de telle sorte que pour une séquence $F_1, F_2, ..., F_m$, les fonctions F_i n'affectent pas les fonctions F_j tel que $j > i$. Autrement dit, les fonctions F_j ne dépendent pas des fonctions F_i.

Afin d'introduire cette démarche, analysons l'exemple suivant :
Supposons les fonctions de sortie $X_1 = F(u, X_3)$, $X_2 = F(u, X_1, X_2)$, $X_3 = F(u)$ et $X_4 = F(u, X_3)$. Le graphe de dépendance est donné par la figure 34a et d'après ce graphe nous remarquons qu'il est possible de choisir une des deux séquences indiqués dans la figure 34b, car X_1 et X_4 ne dépendent pas l'un d'autre.

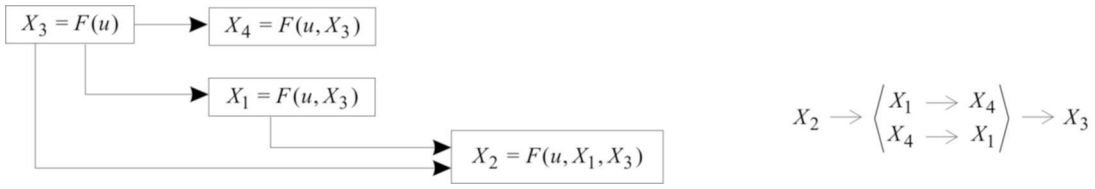


Figure 34 *a)Graphe de dépendance b) Séquences de calcul possibles*

En effet, il s'avère que pour le premier cas ($X_2 - X_1 - X_4 - X_3$) :

1) il est possible de choisir d'abord X_2, car X_1, X_3 et X_4 ne dépendent pas de X_2, c'est-à-dire si X_2 est calculé avant, la nouvelle valeur de X_2 ne changera pas les valeurs attendues dans le calcul de X_1, X_4 et X_3 ;

2) En suivant le même principe, nous constatons qu'il est possible de choisir X_1, car X_3 et X_4 ne dépendent pas de X_1 ;

3) Il est possible de choisir X_4, car X_3 ne dépende pas de X_4; et

4) Enfin X_3 car il est indépendant de X_2, X_1 et X_4.

Nous pouvons suivre le même raisonnement pour le deuxième cas.

Remarque :

Nous soulignons que les fonctions qui sont indépendantes entre elles respectent toujours l'insensibilité à l'ordre. Par conséquent il n'existe aucune contrainte d'ordonnancement pour ces types de fonctions.

☞ Nous ferons la vérification en utilisant des graphes de dépendances.

Nous remarquons que nous n'avons pas prévu une méthode de correction de cette propriété.

Nous utiliserons cette vérification dans l'exemple 3 au chapitre 8. Les exemples 1 et 2 sont insensibles à l'ordre car les fonctions de sorties sont toutes indépendantes entre elles.

UNITE-STABILITE [Yed_00b]

L'unité-stabilité est une propriété qui n'est demandée à vérifier que pour une implantation sur un calculateur qui exécute ces fonctions séquentiellement. Pour un tel calculateur, suite à un signal d'entrée donné qui provoque un changement d'état, l'unité-stabilité demande de rester dans cet état si le signal d'entrée ne change pas.

Pour un système incomplet il n'est pas possible de garantir par avance qu'il soit unité stable. Cependant, il est possible de vérifier que l'évolution de ce système respecte l'unité-stabilité si le système est cohérent. Ainsi, en considérant f_{xi}, g_{xi} et h_{xi} les variables qui définissent un système cohérent et incomplet de chaque sortie X_i, l'expression suivante, nommée hyp_{us}, doit être égale à 0* pour tous les états accessibles ($X_1 \cdot X_2 \quad X_p$).

☞
- La condition $hyp_{us} = 0*$ suivante :

$$hyp_{us} = U_0 \cdot fouh_{etat} \cdot \overline{\left(fouh_{etat} \middle| \left(\prod_{i \in \{1,p\}} X_i \right) = 1* \right)} \qquad \text{où} \quad fouh_{etat} = \prod_{i \in \{1,p\}} (f_{xi} + h_{xi} \cdot X_i)$$

est nécessaire et suffisante pour garantir l'unité-stabilité d'un système complet, mais n'est que suffisante pour un système incomplet (démonstration en annexe E).

- Pour vérifier le système complété, calculer hyp_{us} en considérant :

$$fouh_{etat} = \prod_{i \in \{1,p\}} (F_{xi} + H_{xi} \cdot X_i)$$

En conséquence, si l'implantation est prévue sur un calculateur qui exécute ces fonctions séquentiellement et il s'avère qu'après avoir complété le système, il existe une combinaison avec laquelle *le système n'est pas unité-stable alors l'utilisateur devra modifier la spécification*. Nous proposons l'algorithme 2 (en 6.4.2) en suivant Q5-R2 pour la vérification/correction de cette propriété.

Remarque :

Les systèmes où les fonctions sont cohérentes et indépendantes entre-elles sont unité-stables.

6.2.2 Propriété intrinsèque : Etats interdits

La SF-T4 : $F(X) = 0*$, définit la restriction d'un comportement faisant intervenir plusieurs sorties : par exemple l'assertion A4 de l'exemple 2, $Ouvrir \cdot Fermer = 0*$ interdit que ces deux actions se déroulent au même temps.

Nous établirons plus bas l'expression à respecter afin de garantir que la SF des sorties ne provoquera pas que le système évolue vers un état interdit. C'est-à-dire, qu'à la sortie du calculateur ces états ne sont jamais accessibles.

Expression de vérification

Pour un système incomplet il n'est pas possible de garantir qu'un état à p sorties $X_1 \cdot X_2 \quad X_p$ n'arrivera jamais. Cependant, il est possible de vérifier que l'évolution de ce système ne provoquera l'arrivée dans un état interdit. Cette vérification ne peut être garantie que si le système est cohérent et si et seulement si l'état initial n'est pas dans l'ensemble des états à interdire. En considérant f_{xi}, g_{xi} et h_{xi} les variables qui définissent un système cohérent ($f_{xi} \cdot g_{xi} + f_{xi} \cdot h_{xi} + g_{xi} \cdot h_{xi} = 0*$) et incomplet ($\overline{f_{xi}} \cdot \overline{g_{xi}} \cdot \overline{h_{xi}} \neq 0*$) d'une sortie X_i ; l'expression suivante, nommée hyp_{fm} doit être égal à 0*. En effet,

☞ • La condition $hyp_{fm} = 0*$:

$$hyp_{fm} = \left(\prod_{i \in \{1,p\}} (f_{xj} + h_{xj} \cdot X_j) \right) \cdot \overline{\left(\prod_{i \in \{1,p\}} (f_{xj} + h_{xj} \cdot X_j) \right) \cdot \left(\prod_{i \in \{1,p\}} X_i \right)} = 0*$$

est nécessaire et suffisante pour garantir l'état interdit $X_1 \cdot X_2 \quad X_p$, d'un système complet, mais n'est que suffisante pour garantir l'interdiction de cet état pour un système incomplètement spécifié (démonstration en annexe E).

• Pour vérifier le système complété, calculer hyp_{fm} en considérant F_{xi}, G_{xi} et H_{xi} au lieu de f_{xi}, g_{xi} et h_{xi}

Corrections.

S'il s'avère qu'il existe une combinaison avec laquelle il est possible d'accéder à un état interdit ($hyp_{fm} \neq 0^*$), *l'utilisateur devra changer la spécification* car il n'y a pas de solution alors l'utilisateur devra donc modifier la SF-T4 s'il le considère pertinent ou modifier les spécifications si la SF-T4 doit être respectée. Dans le deuxième cas, nous proposons l'algorithme 2 (en 6.4.2) en suivant Q5-R2 pour la vérification/correction de cette propriété. La vérification/correction devra être faite pour les systèmes de chaque sortie ($X_1, X_2, \quad , X_p$) qui définissent l'état interdit.

REMARQUE :

La dernière expression (hyp_{fm}) est calculée pour des systèmes critiques où l'interdiction à l'état doit être respectée même en cas de panne de capteurs. Pour des systèmes non critiques il est possible de considérer la SF-T3 en multipliant hyp_{fm} par $\overline{U_0}$.

Nous venons de présenter les propriétés à respecter et les expressions à utiliser pour vérifier l'accomplissement de ces propriétés individuellement. Nous soulignons que la séquence de vérification des propriétés est cependant très importante car les corrections demandent d'une part d'enlever des assertions contraignantes et d'autre part d'ajouter des assertions manquantes. Les choix pour faire les corrections, ne peuvent pas être faites dans n'importe quel ordre car on peut nuire à la résolution ou même arriver à un système sans solution. De plus, la résolution de l'incomplétude doit considérer aussi le respect des propriétés. Nous allons détailler ensuite la méthode proposée en prenant en compte ces considérations.

6.3 Méthode de résolution

Les quatre étapes de la méthode, introduite dans 6.1, sont détaillées dans cette section ainsi qu'il est montré dans la figure 35. Ces étapes seront expliquées à l'aide d'exemples dans les chapitres suivants.

Nous proposerons la séquence de la figure 35 car un choix fait par l'utilisateur pour faire respecter une propriété individuellement a cependant une influence sur le respect d'autres propriétés. En effet, les corrections, pour faire respecter les propriétés, demandent d'une part d'enlever des assertions contraignantes et d'autre part d'ajouter des assertions manquantes.

Nous déconseillons de regrouper toutes les contraintes afin de pouvoir identifier les spécifications posant problème. Sinon l'utilisateur n'aurait aucun point de repère pour décider la suite.

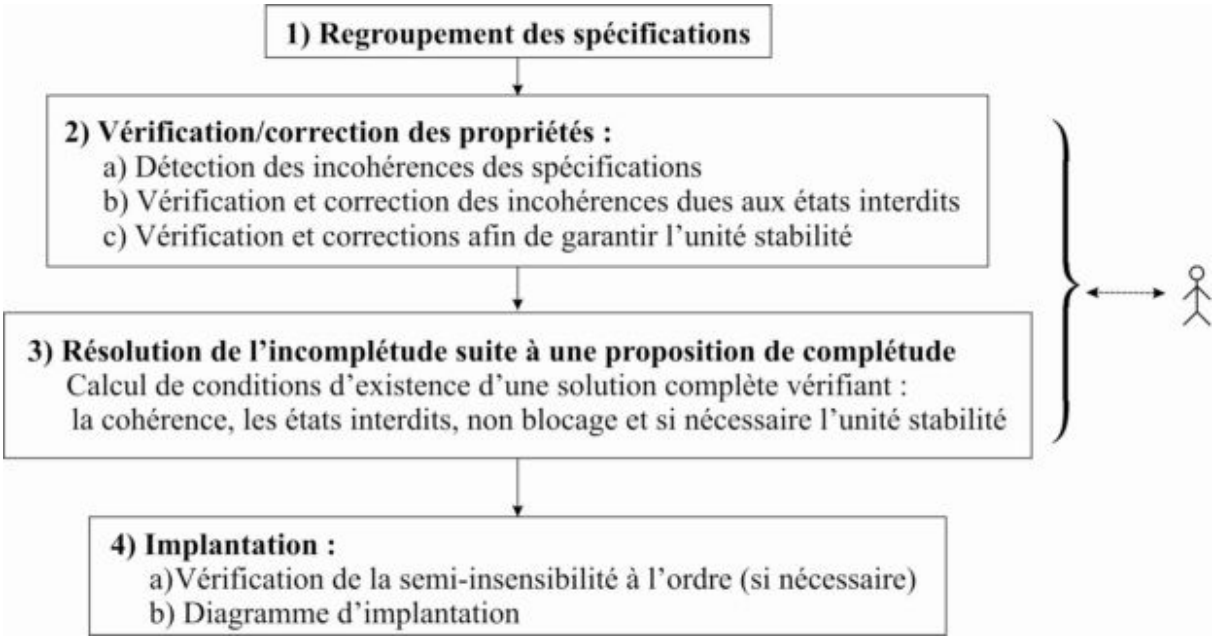


Figure 35 *Méthode de conception formelle de la commande*

Remarques :

1) Notons que la vérification et la correction des propriétés sont faites tout au long des trois dernières étapes de la méthode.
1) Nous soulignons le fait que nous ferons référence à l'utilisation de trois algorithmes, lesquels sont donnés dans 6.4. Ces algorithmes servent à vérifier les propriétés, par rapport aux expressions données dans 6.2, et de les corriger, si nécessaire, de façon individuelle.

6.3.1 Regroupement de la SF

Nous proposons une démarche en sept pas afin de synthétiser les équations et les regrouper en obtenant les deux groupes de systèmes et les deux relations présentés en 6.1. La démarche est mise dans l'annexe E, car le but principal de cette démarche n'est que d'expliquer la suite.

Nous soulignons dans la démarche les deux points suivants :

1) Nous supposons que toutes les spécifications peuvent se ramener, en utilisant des théorèmes, aux 5 types de spécifications présentés dans les chapitres précédents.

   Par exemple (en utilisant [2.25] et [2.37] $\Leftrightarrow$ [2.34]) :

$$F_1(u)\cdot F_2(X)+F_3(u)\cdot F_4(X)=0^* \quad \Leftrightarrow \quad \begin{cases} F_1(u)\cdot F_2(X)=0^* \\ F_3(u)\cdot F_4(X)=0^* \end{cases} \quad \Leftrightarrow \quad \begin{cases} F_1(u)\le \overline{F_2(X)} \\ F_3(u)\le \overline{F_4(X)} \end{cases}$$

2) La décomposition des SF-T2 qui ont un état d'arrivée : $F(u,X)\le Z_j$, en utilisant [2.47] afin de les exprimer comme des SF-T1 parce que nous recherchons la résolution des systèmes élémentaires des signaux de sorties [Med_03].

Par exemple, l'assertion A5 de l'exemple 3 :

$$P\_depose \cdot P\_haute \leq \overline{Avancer} \cdot Descendre \quad \Leftrightarrow \quad \begin{cases} P\_depose \cdot P\_haute \leq \overline{Avancer} \\ P\_depose \cdot P\_haute \leq Descendre \end{cases}$$

Remarque :

Nous décomposons la spécification avec l'objectif d'éviter de faire la composition des sorties pour prouver les propriétés du comportement global. En effet, l'idée de base de notre méthode est de décomposer les propriétés en les exprimant par rapport aux sorties plutôt que de composer les sorties pour vérifier le comportement global du système. Nous soulignons que les expressions de vérification ont été définies ainsi. Nous nous proposons d'éviter le problème de l'explosion combinatoire qu'existe dans les méthodes basées dans la Théorie de la Commande Supervisée.

6.3.2 Vérification et corrections des propriétés

Dans cette étape, de vérification et correction des propriétés, nous ferons une synthèse afin de vérifier la cohérence de la SF par rapport aux propriétés. Nous ne faisons dans cette étape que la vérification des propriétés qui puissent être vérifiées pour des systèmes incomplètement spécifiés : la cohérence, les états interdits et l'unité stabilité.

COHÉRENCE

Vérifier/corriger hyp_1 (système incomplets) en enlevant ou modifiant des assertions. Utiliser l'algorithme 1 pour les systèmes de la forme A et B.

ÉTATS INTERDITS

Il existe deux options de correction pour la relation d'égalité (4) concernant les états interdits de systèmes incomplets. Supposons l'état interdit $X_1 \cdot X_2 \quad X_p$:

- Si après analyse de la part de l'utilisateur la relation $hyp_{fm} \cdot (X_1 \cdot X_2 \quad X_p)$ doit être toujours fausse alors utiliser l'algorithme 2 en considérant la procédure Q5-R2.
- Si après analyse de la part de l'utilisateur l'expression $hyp_{fm} \cdot (X_1 \cdot X_2 \quad X_p)$ n'est pas en réalité toujours fausse, il faut ajouter alors une assertion tel que
  $hyp_{fm} \cdot (X_1 \cdot X_2 \quad X_p) \leq Etat\_non\_interdit$ où $Etat\_non\_interdit \leq \overline{(X_1 \cdot X_2 \quad X_p)}$

UNITÉ-STABILITÉ

Vérifier/corriger hyp_{us} (système incomplets) en enlevant ou en modifiant des assertions. Utiliser l'algorithme 2 en considérant la procédure Q5-R3 pour tous les états non interdits.

Remarque :

Le comportement global d'un système dépend des systèmes de sortie et vice-versa. Les systèmes de comportement global doivent rester cohérents avec les systèmes des sorties suite à une proposition de changement de la SF pour la correction d'une propriété du comportement global tels que l'état interdit et l'unité stabilité.

Nous proposons l'algorithme 2 (6.4.2) en considérant la procédure Q5-R2 et Q5-R3 qui prennent en compte cette problématique. Dans l'exemple 3 nous montrons cette problématique.

6.3.3 Résolution de l'incomplétude

Le système de chaque sortie doit vérifier la complétude. L'expression à vérifier est définie par hyp_4. Nous partons du principe que chaque système de sortie X_i (vérifié et corrigé dans la section précédente) peut être complété avec des équations faisant intervenir les paramètres k_{fxi}, k_{gxi} et k_{hxi}. Il est exigé cependant que les choix faits pour compléter le système respectent aussi toutes les propriétés de 6.2. En effet, les choix de ces paramètres sont restreints par la vérification des propriétés et en plus par le mode d'exécution du contrôleur s'il exécute les fonctions de contrôle en séquentiel.

Bien qu'il est possible de faire la vérification de la somme de toutes ces contraintes, nous ferons la vérification de chaque propriété individuellement afin de faire connaître à l'utilisateur quel est la cause du non respect d'une spécification et qu'il puisse réagir en conséquence pour corriger la spécification.

En considérant que l'utilisateur, après une analyse de l'incomplétude, peut ajouter des assertions à cause d'oublis dans la SF nous proposons ci-dessous une démarche de vérification après l'ajout d'une ou plusieurs assertions.

Démarche de vérification/correction :

La propriété à vérifier est hyp_4 du système calculé dans la section précédent. Pour la vérification/correction de cette propriété nous proposons **l'algorithme 2** (en 6.4.2) en suivant d'abord l'option Q4-R3[2] .

Si le système analysé n'est pas complet alors suivre l'option Q5-R2 après avoir reçu la réponse de l'utilisateur, c'est-à-dire k_{fxi} , k_{gxi} et k_{hxi} . Cette option validera que les assertions introduites par l'utilisateur vérifient les autres propriétés avant de continuer.

L'option **Q5-R2** de cet algorithme considère la séquence de vérification des propriétés suivante :

A) *Non-blocage* : Vérifier cette propriété pour le système de chaque système des sorties X_i (voir 6.2.1). En cas de besoin utiliser l'algorithme 3 pour ajouter des assertions. Si la SF a été modifiée suite à un ajout d'assertion, recommencer la résolution (6.3.3).

B) *Cohérence* : Vérifier/corriger cette propriété pour le système de chaque système des sorties X_i, en enlevant ou modifiant des assertions..

C) *États interdits* : Calcul de la condition d'existence du système cohérent respectant les états interdits (hyp_{fm}). En cas de non respect suivre la même procédure du 6.3.2 pour les états interdits et recommencer la résolution (6.3.3), car la SF a été modifiée.

D) *Unité-stabilité* : Pour une implantation prévue en exécution séquentielle, calculer la condition d'existence du système cohérent respectant l'unité-stabilité (hyp_{us}).En cas de non respect suivre la même procédure du 6.3.2 pour l'unité stabilité et recommencer la résolution (6.3.3), car la SF a été modifiée.

E) Continuer l'algorithme 2 afin de vérifier si l'incomplétude est enlevée.

Remarques :

1) Si $k_f + k_g + k_h = hyp_4$, utiliser les expressions calculées pour un système complet.

2) Dans l'exemple 2 et 3, nous choisirons de laisser toute l'incomplétude (hyp_4) dans l'état précédant (mise en mémoire) : $k_f = 0*$, $k_g = 0*$ et $k_h = hyp_4$. Ceci afin de présenter la

[2] L'option **Q4-R3** de cet algorithme restreint les questions à l'utilisateur à $hyp_4 \cdot \overline{U_0}$ afin de ne poser des questions que pour les signaux que puisse arriver

démarche pour le point 1 de cette remarque et en plus, pour montrer que, pour des cas d'étude, une démarche de résolution peut être systématisée.

3) Nous précisons cependant que dans la pratique la systématisation d'une réponse tels que la mise à 1, à 0 ou en mémoire de toute l'incomplétude, doit se prendre avec mesure. Nous le mettons en évidence dans la résolution de l'exemple 1 et 2.

CALCUL DES FONCTIONS DE SORTIES :

Une fois que tous les systèmes de sorties sont complets et que toutes les propriétés sont respectées il est possible de calculer la solution des fonctions de sortie. Ces fonctions de sorties est exprimé par un SR ou un RS comme suit :

✍ $X_i = SR(F_{xi}, G_{xi})$ ou bien $X_i = RS(F_{xi}, G_{xi})$

6.3.4 Calcul et implantation de la fonction des sorties

L'implantation de la fonction de contrôle peut se réaliser sur un calculateur exécutant les fonctions de contrôle en parallèle ou séquentiel (voir la figure 36). Si les fonctions des sorties sont indépendantes les unes par rapport aux autres alors l'implantation sur les deux types de calculateurs est possible car la fonction à implanter F_i est alors égale à la fonction de chaque sortie X_i dans les deux calculateurs.

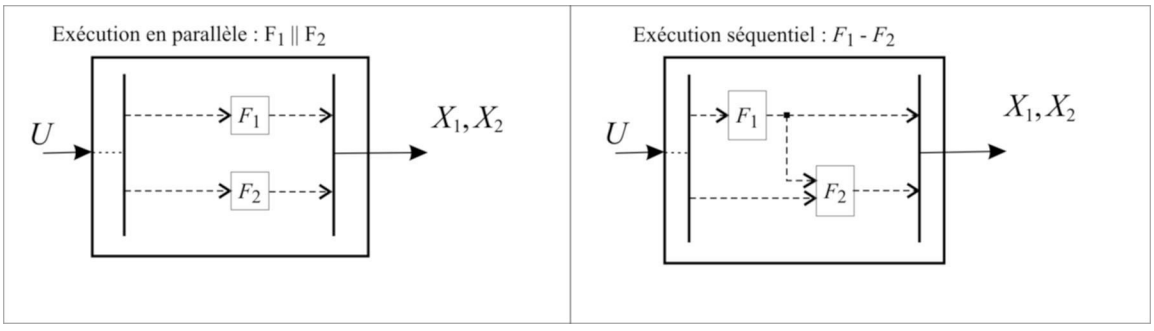


Figure 36 *Implantation de la fonction de contrôle, exécuté en :* a) *parallèle (par exemple en logique câblée)* b) *séquentielle (par exemple en programme ladder).*

Le calculateur qui exécute les fonctions de contrôle en séquentiel est utilisé couramment dans l'industrie et les fonctions des sorties à implanter sur ces calculateurs sont représentées par les opérations SR et RS. Rappelons cependant que la fonction à implanter sur un calculateur séquentiel, n'est pas forcément la même fonction que la fonction de sortie du calculateur. Or,

si la fonction de sortie respecte la semi-insensibilité à l'ordre alors la même fonction de sortie peut être implantée sur le calculateur séquentiel en respectant le même ordre dans lequel a été prouvé la semi-insensibilité à l'ordre.

La vérification et la correction de l'unité-stabilité ont été introduites dans la section précédente (point D) mais pas la semi-insensibilité à l'ordre parce que :

1) Pour un calculateur exécutant les fonctions de contrôle en séquentiel, il faut établir au moins une séquence d'exécution afin de vérifier la semi-insensibilité à l'ordre.

2) Nous n'avons pas encore une expression pour traduire cette propriété afin de garantir en avance le respect de la SF ni pour un système incomplet ni pour un système complet.

Nous nous limitons aux fonctions de sorties dont l'ordonnancement peut être déduit à partir d'un graphe de dépendances (voir 6.2.1). Nous précisons que ce graphe doit être construit à partir du système de sorties.

Enfin, les mêmes fonctions des sorties calculées en 6.3.3 et représentées par les opérations SR ou RS peuvent être implantées sur le calculateur exécutant les fonctions de contrôle en séquentiel.

D'autre part, pour un calculateur exécutant les fonctions de contrôle en parallèle, la fonction calculée peut être implanté directement en utilisant les variables F_{xi}, G_{xi} ou bien H_{xi}. Nous allons donner à chaque exemple, l'implantation sur un tel calculateur. La fonction à implanter sur un tel calculateur peut être exprimée par une équation récurrente, tel que :

✍ $$X_i = (F_{xi} + k_{fxi}) + \overline{\left(G_{xi} + k_{gxi}\right)} \cdot x_i = (F_{xi} + k_{fxi}) + (\overline{G_{xi}} \cdot \overline{k_{gxi}}) \cdot x_i$$

6.4 Algorithmes de vérification des propriétés

A continuation nous allons donner les trois algorithmes de vérification/correction des propriétés selon la caractéristique de la propriété à vérifier (égale à 0* ou différent de 0*) et l'action à suivre pour faire la correction (ajouter ou enlever d'assertions). Le but de présenter ainsi les algorithmes est de permettre l'inclusion dans la méthode de nouvelles propriétés qui puisse être définies plus tard.

Ces algorithmes servent en premier instance comme *un outil de vérification des propriétés* au cours de la synthèse. En cas d'un non respect de la propriété, une démarche est proposée afin de surmonter le non respect de la propriété à l'aide de l'utilisateur.

6.4.1 L'algorithme 1 : $hyp_v = 0*$, correction par l'enlèvement des contraintes.

Cet algorithme (voir figure 37) représente les pas à suivre pour vérifier et, si nécessaire, corriger une SF contraignante afin de respecter la propriété car la recherche de solutions pour le système d'équations qui la représente oblige, pour établir la forme générale du signal de sortie solution, d'introduire des hypothèses restrictives non admissibles sur le comportement des signaux d'entrée. Dans cet algorithme il est prévu la communication avec l'utilisateur en lui posant des questions précises afin que le système soit modifié par l'utilisateur lui-même. La sortie de l'algorithme est le système vérifié ou bien, en cas d'échec et sans réponse de l'utilisateur, la sortie serait le message de non accomplissement de la propriété en question. La démarche de cet algorithme sera détaillée dans le premier exemple de ce chapitre.

Avant d'expliquer l'algorithme, nous précisons que si la SF comporte des spécifications de fonctionnement du type 3 (SF-T3), alors l'introduction de cette information s'impose (E_{u0_1}) et deux pas additionnels sont ajoutés (P_{u0_1} et P_{u0_2}) qui sont signalés en vert sur la figure. Ceci est la conséquence de que la SF-T3 comporte des informations sur les entrées qui ne sont pas sensés à arriver dans un fonctionnement normal et les pannes ne sont pas prévues. Dans ce cas là, l'information de la SF-T3 peut être enlevé de l'algorithme de vérification parce que :

Si $a = 0*$ et $a \leq X$ alors nous avons $0* \leq X$ et ceci est toujours vrais ([2.31]).

La démarche de l'algorithme est la suivante :

Le premier pas à faire c'est lire la SF-T3 (E_{u0_1}), ensuite enlever de la vérification ces cas de figure (P_{u0_1}) car U0 est déjà égal à 0*. Ensuite, calculer la propriété (P1) et vérifier si la propriété est accomplie (Q1), si ce le cas l'algorithme finira en donnant comme réponse le même système (P2) comme sortie vérifiée (S3) car le nombre de vérifications nv =1 (Q3). D'autre part, en cas d'échec (Q1), il est prévue de calculer la partie du système qui accomplie la propriété (P3). Après, deux options sont prévues en avance (Q4), l'arrête de l'algorithme (Q4-R1) et demander à l'utilisateur de modifier le système (Q4-R2) en enlevant des assertions qui font que le système soit très contraignant. Ensuite, nous analysons l'information donnée par l'utilisateur si la réponse n'apporte aucune nouvelle information (Q2), il faut revenir à faire le choix (Q4). En cas où, il apporte une nouvelle information alors nous complémentons la SF-T3 (P_{u0_2}), sinon nous prenons en compte ces informations (P4) pour relancer une nouvelle vérification (nv=2) avec ce nouveau sous-système : f_{av}, g_{av} et h_{av} (P5). Nous continuerons la vérification le nombre de cycles de vérifications (nv) que soient nécessaires afin que l'utilisateur arrive à surmonter le non accomplissement de la propriété en revenant par le point

2 (au moins qu'il décide de ne pas donner plus d'informations sans avoir surmonté le non accomplissement de la propriété en sortant par R1 à la question Q4). Si le non accomplissement est surmonté alors la nouvelle fonction modifiée (P6) sera renvoyée comme sortie de l'algorithme à l'utilisateur (S3).

Nous soulignons, que cet algorithme permet une démarche évolutive dans lequel l'utilisateur afin d'enlever les contraintes (R2) donne des nouvelles assertions. Pourtant, celles-ci n'obligent pas à refaire la vérification de tout le système d'équations, sinon uniquement de la partie ajoutée par l'utilisateur (P4).

6.4.2 L'algorithme 2 : $hyp_v = 0*$, correction par l'ajout des assertions.

Cet algorithme, montré dans la figure 38, représente les pas à suivre pour corriger une SF afin de respecter la propriété en ajoutant des assertions au système. Ces corrections ne peuvent pas être faites que si le système est incomplet. Dans cet algorithme, de la même manière que l'algorithme précédent, il est prévu la communication avec l'utilisateur en lui posant des questions précises afin que le système soit modifié par l'utilisateur lui-même et que les propriétés soient respectées. La sortie de l'algorithme est le système vérifié ou bien, en cas d'échec et sans réponse de l'utilisateur, la sortie serait le message de non accomplissement de la propriété en question. Cet algorithme sera appliqué dans les exemples.

Avant d'expliquer l'algorithme, nous précisons que si la SF comporte des spécifications de fonctionnement du type 3 (SF-T3), alors l'introduction de cette information, n'est pas considérée car la propriété doit être prouvé en considérant U0.

La démarche de l'algorithme est très semblable à l'antérieur à différence près des points suivants : Il n'est pas considéré les SF-T3 car la propriété doit être prouvé en considérant U0. Ainsi, les pas E_{u0_1}, P_{u0_1} et P_{u0_2}, ne sont calculés. La question (Q1) est la même : si la réponse est positive alors la sortie de l'algorithme est prévue (P2, Q et S3).

Dans le cas contraire, où la propriété n'est pas accomplie alors la question au concepteur n'est pas d'enlever de contraintes en choisissant une option comme dans le dernier algorithme. Il est demandé (Q4), en contrepartie, à l'utilisateur de donner ses choix sur la partie qui n'est pas encore choisit pour faire accomplir la propriété (R2). S'il décide de ne pas donner aucune assertion alors l'algorithme s'arrête (R1), sinon une fois reçue la réponse non nulle (Q2) alors utiliser Q5-R1 et passer directement à (P4) car il ne reste qu'à vérifier que le système avec la nouvelle assertion reçue accompli la propriété (P5).

Remarques :

1) Les pas à suivre pour **Q5-R2** ont été définis dans le chapitre 6 et cette option sera utilisée pour la résolution de l'incomplétude.
2) Nous avançons que pour résoudre la complétude une option particulière est prévue (**Q4-R3**) afin de ne demander à l'utilisateur que des informations sur des entrées qui puissent arriver. C'est algorithme est définie et utilisé dans l'exemple 1 (voir 7.1.3)

Procédure [Q5-R2] de l'algorithme 2 :

Cette partie de l'algorithme 2 a été définit pour simplifier les corrections de la complétude. Comme U_0 n'arrive jamais, ça ne vaut pas la peine de demander à l'utilisateur de faire un choix car *n'importe quel réponse de l'utilisateur n'aura aucun effet sur le comportement du système en absence de défaillances*. Nous considérons donc les pas suivants afin de considérer U_0 dans la résolution de l'incomplétude (voir 6.3.2.2) :

1) enlever U0 de la question à l'utilisateur : $hyp'_v = hyp_v \cdot \overline{U_0}$
2) Si $hyp'_v \neq 0*$, continuer au Q4-R2 de l'algorithme 2, en posant la question à l'utilisateur
3) Si $hyp'_v = 0*$ alors finir l'algorithme 2.

Procédure Q5-R2 de l'algorithme 2 :

Cette partie de l'algorithme 2 a été définit pour la résolution de l'incomplétude (voir 6.3.3)

A) *Non-**Blocage*** : Si une sortie est bloquante (6.2.1) utiliser l'algorithme 3 pour ajouter d'assertions si le système de cette sortie est incomplet. Si le système est bloquant et complet alors il n'y a pas de solution car le système est très contraignant. Si la SF a été modifiée suite à un ajout d'assertion, recommencer la résolution (6.3.3).

B) ***Cohérence*** : Calcul de la condition d'existence du système de sorties cohérent ($hyp_1 = 0*$). En cas de non respect, faire les corrections nécessaires en utilisant l'algorithme 1.

C) ***États interdits*** : Calcul de la condition d'existence du système cohérent respectant les états interdits (hyp_{fm}). En cas de non respect suivre la procédure du 6.3.2 pour les états interdits et recommencer la résolution (6.3.3), car la SF a été modifiée.

D) ***Unité-stabilité*** : Pour une implantation prévue en exécution séquentielle, calculer le respect de l'unité-stabilité (hyp_{us}) du système cohérent. En cas de non respect suivre la procédure du 6.3.2 pour l'unité-stabilité et recommencer la résolution (6.3.3), car la SF a été modifiée.

E) Continuer l'algorithme 2 au **P4** afin de vérifier si l'incomplétude est enlevée.

6.4.3 L'algorithme 3 : $hyp_v \neq 0*$ par l'ajout des assertions

Cet algorithme, montré dans la figure 39, représente les pas à suivre pour corriger une SF afin de respecter la propriété en ajoutant des assertions au système. Ces corrections ne peuvent pas être faites que si le système est incomplet. Dans cet algorithme, de la même manière que l'algorithme 1, il est prévu la communication avec l'utilisateur en lui posant des questions précises afin que le système soit modifié par l'utilisateur lui-même et que les propriétés soient respectées. La sortie de l'algorithme est le système vérifié ou bien, en cas d'échec et sans réponse de l'utilisateur, la sortie serait le message de non accomplissement de la propriété en question. Cet algorithme sera appliqué dans les exemples.

Avant d'expliquer l'algorithme, nous précisons que si la SF comporte des spécifications de fonctionnement du type 3 (SF-T3), alors l'introduction de cette information s'impose (E_{u0_1}) et deux pas additionnels sont ajoutés (P_{u0_1} et P_{u0_2}) qui sont signalés en vert sur la figure. Ceci est dû au fait que le calcul théorique de la propriété pourrait donner une fausse information par rapport au système physique. En effet, la vérification théorique peut donner comme réponse que hyp_2 a une valeur : $hyp_2 \neq 0*$ mais que cette valeur n'arrive jamais dans le système physique (information donné par SF-T3).

La démarche de l'algorithme est très semblable au premier algorithme à différence près des points suivants : Nous recherchons à avoir si *au moins une condition accomplie la propriété*, si ce le cas la sortie de l'algorithme est prévue (Q1), (P2), (Q3) et (S3), en prenant en compte l'information de la SF-T3 (P_{u0_1}). D'autre part, si la propriété n'est pas accomplie (Q1) mais le système est déjà complètement spécifié (Q2) alors il est possible d'assurer que la propriété n'est pas respectée. C'est-à-dire, la fonction de contrôle est déjà définie et la propriété n'est pas respectée. Dans le cas contraire, où la propriété n'est pas accomplie alors la question au concepteur n'est pas d'enlever de contraintes en choisissant une option comme dans le dernier algorithme. Il est demandé (Q4), en contrepartie, à l'utilisateur de donner au moins une assertion parmi l'incomplétude pour faire accomplir la propriété (R2). S'il décide de ne pas donner aucune assertion alors l'algorithme s'arrête (R1), sinon une fois reçue la réponse non nulle (Q2) alors il ne reste qu'à vérifier que la nouvelle assertion reçue fasse accomplir la propriété (P5).

Les diagrammes de flux des algorithmes proposés :

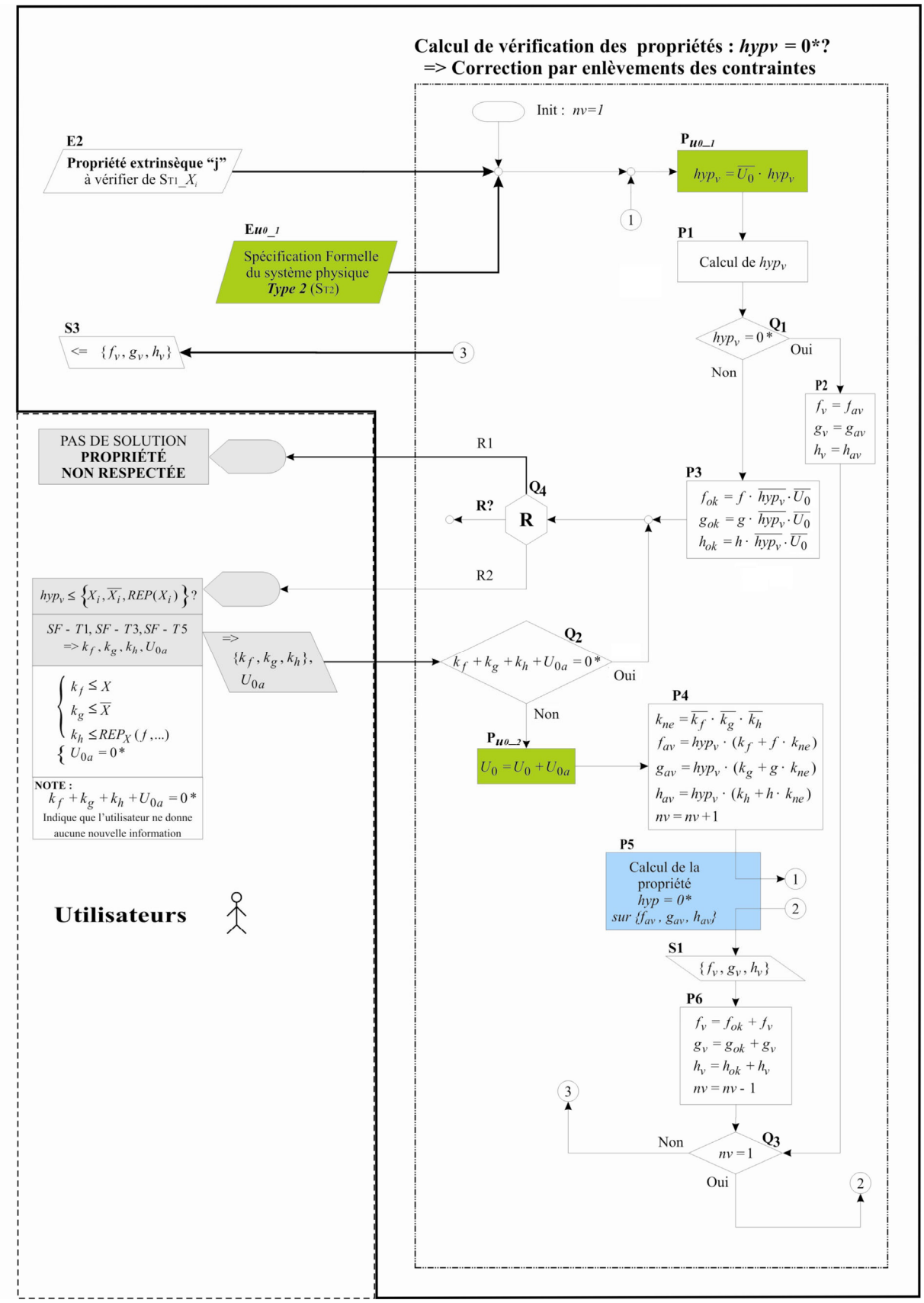


Figure 37 *Algorithme 1, vérification de la propriété : hyp = 0* et corrections par enlèvements des contraintes*

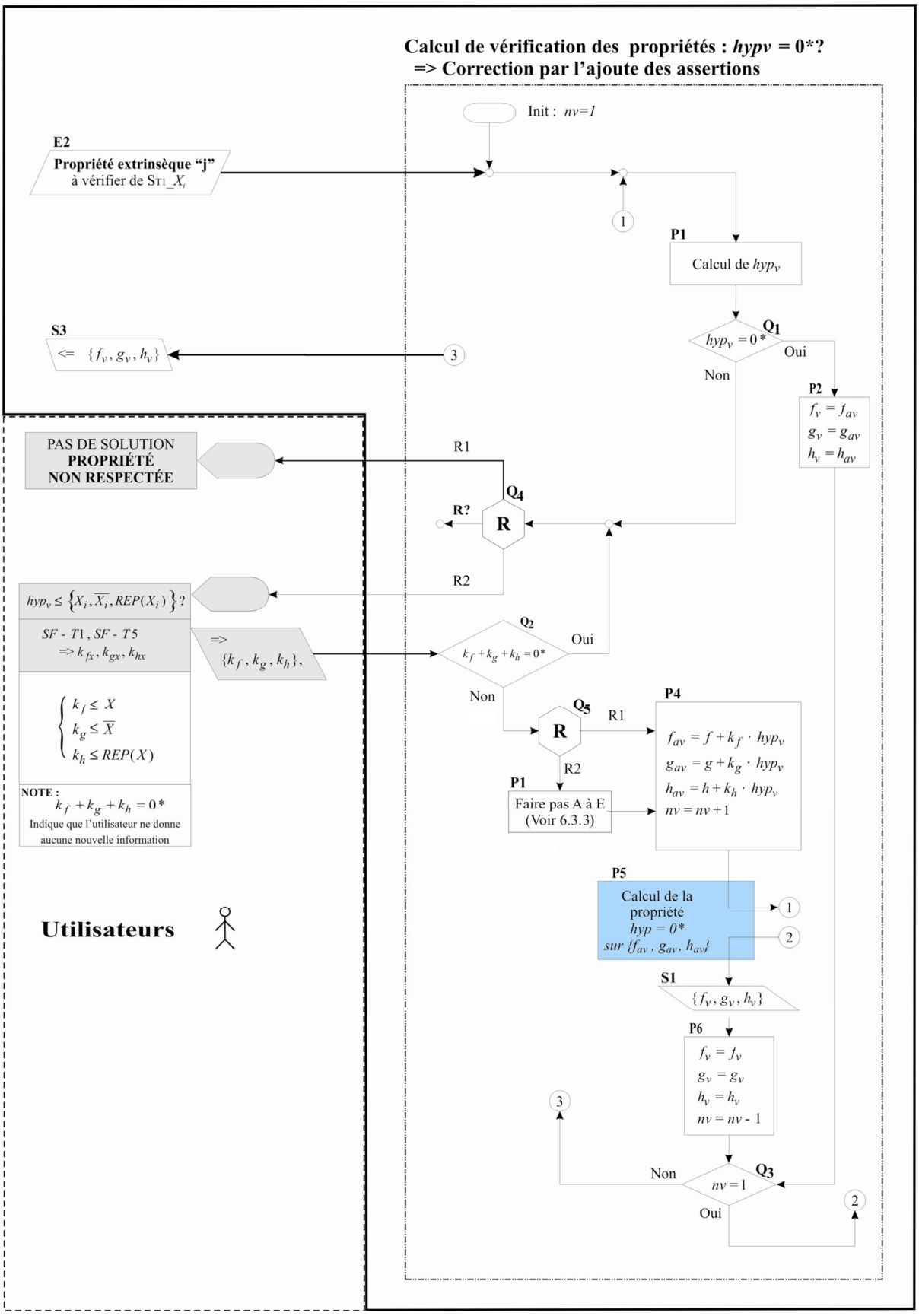


Figure 38 *Algorithme 2, vérification de la propriété : hyp = 0* et corrections par l'ajoute des assertions*

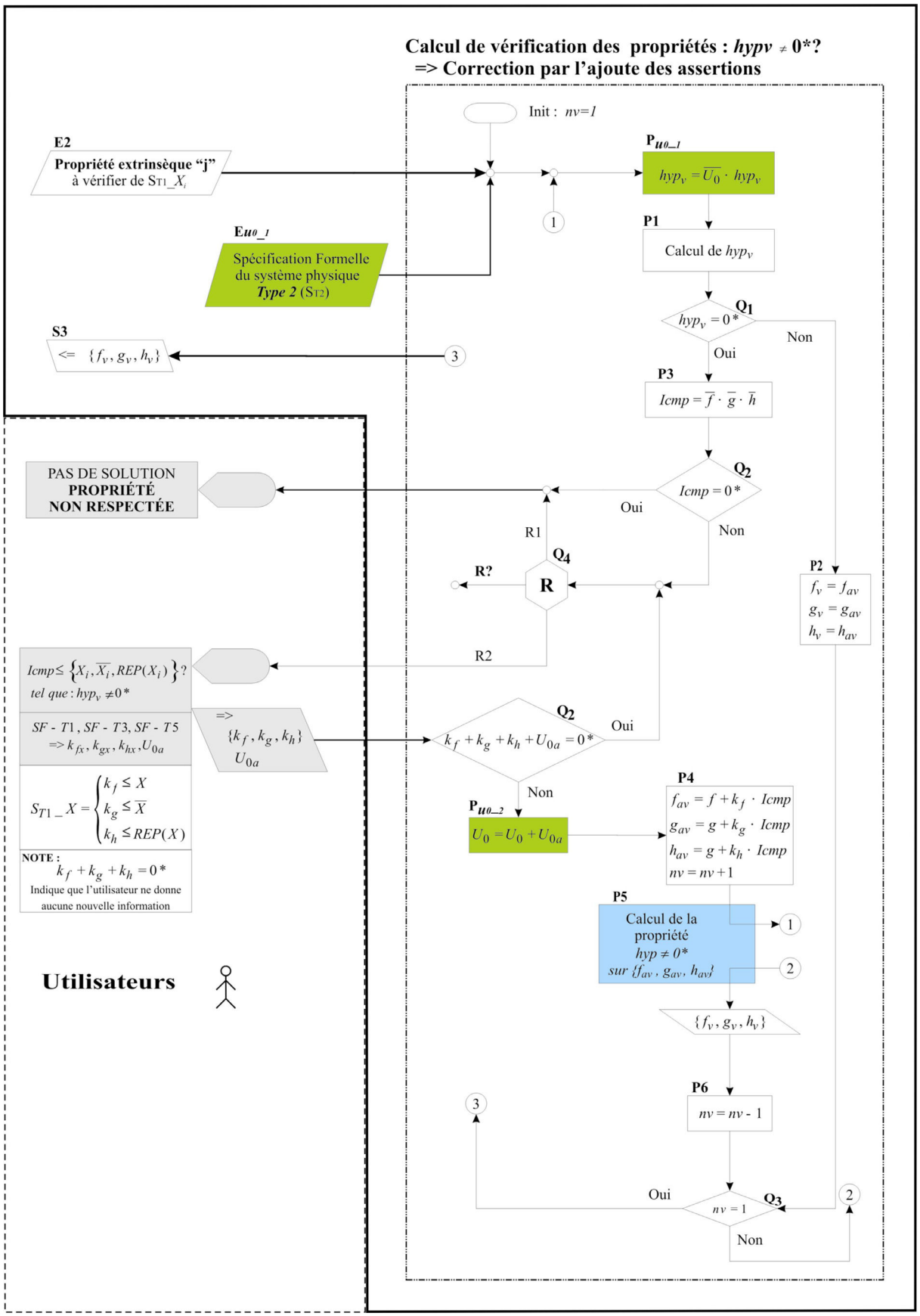


Figure 39 *Algorithme 3, vérification de la propriété : hyp ≠ 0* et corrections par l'ajoute des assertions*

6.5 Conclusion

La méthode a été développée afin de permettre de résoudre le plus de cas possibles en suivant trois algorithmes généralisés par les propriétés à respecter et la correction proposée. Ces algorithmes indiquent les pas à suivre afin de corriger et de surmonter le non accomplissement des propriétés. Il est possible cependant de cibler ces algorithmes par rapport à la propriété à vérifier et à la phase de vérification (2 ou 3).

Cette méthode prévoit une démarche qui permet à l'utilisateur de donner de nouvelles informations afin d'enlever le non respect des propriétés. Afin d'identifier le mieux possible la cause, nous avons prévu l'analyse individualisée de chaque propriété.

D'ailleurs, nous montrerons qu'il n'est pas toujours nécessaire de suivre la trace d'un système pour garantir le respect de la SF et les propriétés extrinsèques. L'idée de base de cette démarche est de travailler avec un système d'équations par sortie. A partir de ce système d'équations nous allons vérifier/corriger les spécifications de telle sorte à respecter les propriétés demandées.

La clef est de calculer les expressions de vérification des propriétés par rapport aux systèmes de sorties (forme A). Nous voulons ainsi éviter le problème de l'explosion combinatoire de la Théorie de la Commande Supervisée.

Chapitre 7

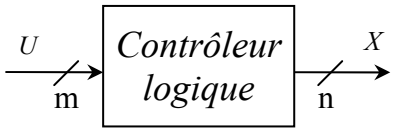
Dans ce chapitre nous présentons la résolution des deux premiers exemples. A travers la résolution, nous illustrons l'intervention de l'utilisateur pour corriger les incohérences dans la SF par rapport aux propriétés attendues.

Les exemples sont introduits par degré de complexité afin de présenter la méthode d'une façon évolutive. En effet, le premier exemple ne considère que des SF-T1 et le deuxième introduit des SF-T3, SF-T4 et en plus une assertion incluant l'opération TON.

Nous allons expliquer la méthode de synthèse, d'abord en utilisant l'exemple 1 (mono-sortie) pour illustrer les principes de la méthode; ensuite nous allons introduire la méthodologie de résolution pour des systèmes multi-sorties avec l'exemple 2 (à deux sorties).

7. Études de cas : Exemple 1 et 2

Nous rappelons que la méthode proposée repose sur la résolution d'un système d'équations écrit en algèbre $\mathbb{I}$ qui représente la SF. Le but est de calculer les fonctions de sorties X_i exprimées par les opérations SR ou RS (voir la figure 40) tout en respectant la SF donnée et les propriétés attendues.



$$X_j = SR(f_1, g_1)$$
ou bien $X_j = RS(f_2, g_2)$
tel que $f_1, f_2, g_1, g_2 = F(U, X)$
X_j : le j ème élément des sorties

Figure 40 *Représentation des entrées-sorties d'un SLS et des équations solutions*

Dans ce chapitre, nous résolvons d'abord le premier exemple de la commande d'un système mono-sortie : le contrôle d'une pompe de remplissage. Le deuxième exemple, résolu ensuite, concerne la commande d'un système à deux sorties : l'ouverture et la fermeture d'un portail.

7.1 Résolution de l'exemple 1 : Système à une sortie

Le système étudié dans cette section a été formalisé dans la section 4.3.1. Le système d'équations formalisé en $\mathbb{I}$ est le suivant :

$$\begin{cases} A1) & Res\_vide \le Pomper \\ A2) & Res\_plein \le \overline{Pomper} \\ A3) & Bas\_vide \le \overline{Pomper} \end{cases}$$

L'objectif est donc d'exprimer la sortie $Pomper$ en utilisant une opération SR ou bien une opération RS, tout en respectant la spécification donnée. Les trois assertions *A1*, *A2* et *A3*, ont été établies manuellement indépendamment les unes des autres. Si leur formalisation sur $\mathbb{I}$ permet de s'affranchir des imprécisions du LN, rien ne garantit que la réunion de ces expressions formelles constitue un système d'équations ayant une solution qui respecte en plus les propriétés extrinsèques.

Nous allons suivre la méthode décrite dans le chapitre précédent mais nous soulignons que pour un système à sorties indépendantes sans SF-T4 :

1) La cohérence garantit la stabilité de chaque sortie individuellement donc le fait de prouver la cohérence des sorties implique en même temps l'unité-stabilité.
2) La vérification de la semi-insensibilité à l'ordre n'est pas nécessaire car aucune restriction d'ordre de calcul n'est nécessaire quand les sorties sont toutes indépendantes entre-elles.

Par conséquent, pour qu'un système n'ayant que des SF-T1 (et SF-T3 d'ailleurs) ait une solution implantable et correctement spécifiée le système d'équations doit respecter les propriétés extrinsèques suivantes : 1) cohérence, 2) complétude, 3) non blocage des sorties.

Ainsi, les pas à suivre pour la résolution d'un système à sorties indépendantes sont donc,

Pas 1) Regroupement de la spécification de fonctionnement (6.3.1).

Pas 2) Vérification et corrections des propriétés :

- Vérification et résolution des incohérences du système de sorties (6.3.2).

Pas 3) Résolution de l'incomplétude (6.3.3) :

- Condition d'existence du système des sorties cohérent et complet.
- Conditions à vérifier afin que les sorties ne soient pas bloquantes.
- Méthodologie de vérification/correction pour la résolution de l'incomplétude.

Pas 4) Implantation (6.3.4) :

- Diagramme d'implantation de la fonction de sortie.

7.1.1 Regroupement de la spécification de fonctionnement

En utilisant l'équation [2.46] nous allons regrouper les assertions *A2* et *A3*, comme suit :

$$\begin{cases} A1 & Res\_vide \le Pomper \\ \begin{cases} A2 & Res\_plein \le \overline{Pomper} \\ A3 & Bas\_vide \le \overline{Pomper} \end{cases} \end{cases} \equiv \begin{cases} A1) & Res\_vide \le Pomper \\ \{A2 \cdot A3) & (Res\_plein + Bas\_vide) \le \overline{Pomper} \end{cases}$$

7.1.2 Vérification et corrections des propriétés

RÉSOLUTION DES INCOHÉRENCES DU SYSTÈME DE SORTIES

Nous suivons l'algorithme 1 pour la vérification et la résolution des incohérences. (6.3.2). D'après [5.1], il est possible de calculer la cohérence de ce système. En effet, en utilisant hyp_1 de [5.1], nous avons (à l'étape P1 du premier cycle de calcul de cet algorithme) :

$$\textbf{P1\_1)} \quad hyp_1 = \begin{pmatrix} Res\_vide \cdot (Res\_plein + Bas\_vide) \\ + Res\_vide \cdot 0^* + \left(Res\_plein + Bas\_vide\right) \cdot 0^* \end{pmatrix}$$
$$= Res\_vide \cdot (Res\_plein + Bas\_vide)$$

Ce résultat n'est pas égale à 0*. **Cela veut dire que le système est incohérent**. C'est à l'utilisateur donc de donner des réponses pour enlever l'incohérence.

ANALYSE PARTIELLE :

En observant le système, il avère que les signaux venant des capteurs Res_vide et Res_plein ne sont jamais vrais au même temps en absence de défaillances de capteurs. En effet, la citerne ne peut pas être pleine (Res_plein) et vide au même temps (Res_vide) en mode normal de fonctionnement. Supposant qu'aucune démarche additionnelle en cas de défaillances n'est prévue par l'utilisateur, nous recevrons alors comme réponse une SF-T3 qui sera intégrée à la SF comme l'assertion suivante :

☞ **REPONSE :** $A4)\quad Res\_vide \cdot Res\_plein = 0*$

En suivant l'algorithme 1, à l'étape P1 du deuxième cycle de calcul, nous obtenons la nouvelle valeur de hyp_1 :

P1_2) $hyp_1 = \overline{Res\_plein} \cdot Res\_vide \cdot Bas\_vide$

Le système reste incohérent car le dernier résultat n'est pas égal à 0*.

ANALYSE PARTIELLE :

En continuant avec le choix de consulter l'utilisateur, nous lui demandons en deuxième instance de nous informer sur l*es modifications à faire à la spécification*. Car, nous n'attendons plus de nouvelles SF-T3 à ajouter. En effet, ***le système est contraint et sans solution autrement dit INCOHERENT***. Ceci peut se mettre en évidence à partir du dernier résultat de hyp_1. En effet, la spécification telle qu'elle est, à ce point de la synthèse, implique le sous système suivant :

$$\begin{cases} Res\_vide \le Pomper \\ (Res\_plein + Bas\_vide) \le \overline{Pomper} \\ Res\_vide \cdot Res\_plein = 0* \end{cases} \Rightarrow \begin{cases} \overline{Res\_plein} \cdot Res\_vide \cdot Bas\_vide \le Pomper \\ \overline{Res\_plein} \cdot Res\_vide \cdot Bas\_vide \le \overline{Pomper} \end{cases}$$

A partir de l'évolution du système physique, il peut être constaté que hyp_1 peut arriver, c'est-à-dire $\exists t_i \mid \left(\overline{Res\_plein} \cdot Res\_vide \cdot Bas\_vide\right)(t_i) = 1$. Cela veut dire qu'aux instants t_i, selon la spécification, l'utilisateur voudrait démarrer la pompe et l'arrêter avec le même signal d'entrée : $\overline{Res\_plein} \cdot Res\_vide \cdot Bas\_vide$. Sans plus d'information, il est impossible de proposer une solution car la cohérence est une condition nécessaire pour avoir une solution.

En suivant avec l'algorithme 1, la question à poser à l'utilisateur (**Q4-R2**), qui ressort tout de suite, est de savoir quoi faire avec la condition d'incohérence (hyp_1) : il faut mettre la pompe en route, l'arrêter ou simplement la laisser dans son état précèdent $\{Pomper, \overline{Pomper}, REP_{Pomper}\}$? ***Il faut donc que l'utilisateur fasse un choix sur ce dernier sous-système incohérent***.

Nous allons supposer pour cet exemple, que son besoin est de protéger la pompe car elle ne doit pas fonctionner à vide. Il décide donc d'introduire l'assertion suivante qui n'a pas été considérée au départ :

☞ **REPONSE :** $\begin{cases} k_f = k_h = 0^*, \quad k_g = hyp_v = \overline{Res\_plein} \cdot Res\_vide \cdot Bas\_vide \end{cases}$

C'est-à-dire, $A5)$ $\overline{Res\_plein} \cdot Res\_vide \cdot Bas\_vide \leq \overline{Pomper}$

De ce fait, en suivant l'algorithme 1, à l'étape P1 du troisième cycle de calcul, nous obtenons la nouvelle valeur de hyp_1 :

P1_2) $hyp_1 = 0^*$

✍ **LE SYSTÈME DE LA SF A ÉTÉ MODIFIÉ AFIN DE RESPECTER LA COHÉRENCE :** $hyp_1 = 0^*$.

En continuant l'algorithme 1 de vérification et de correction de la cohérence nous obtenons ***comme résultat le système, de cette sortie, cohérent suivant avec les modifications faites par l'utilisateur*** :

$$\begin{cases} A1)_{coh} & Res\_vide \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide} \leq Pomper \\ A2 \cdot A3 \cdot A5)_{coh} & \overline{Res\_vide} \cdot Res\_plein + Bas\_vide \cdot \overline{Res\_plein} \leq \overline{Pomper} \\ A4) & Res\_vide \cdot Res\_plein = 0^* \end{cases}$$

7.1.3 Résolution de l'incomplétude

Pour cet exemple, nous allons suivre la démarche où l'utilisateur choisira la partie incomplète de la commande (6.3.1) au fur et à mesure de l'utilisation de l'algorithme 2 pour vérifier et corriger la complétude $\overline{f} \cdot \overline{g} \cdot \overline{h} = 0^*$ en suivant l'option **Q5-R2** de cet algorithme afin de respecter les autres propriétés :

En suivant l'algorithme 2, dans un premier cycle de calcul, nous obtenons pour la vérification de l'incomplétude hyp_4 au **P1** le résultat suivant après simplification en utilisant notre programme développé en *Mathématica* :

P1_1) $hyp_4 = \overline{Res\_vide} \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide} + Res\_plein \cdot Res\_vide$

Cette expression n'est pas égale à 0* et **cela veut dire que le système n'est pas complètement spécifié (incomplète)**.

En suivant la démarche prévue, nous suivons l'option Q4-R3 afin de ne pas inclure U_0 dans la question à l'utilisateur :

PROCÉDURE [Q4-R3] DE L'ALGORITHME 2 :

1) $$\begin{aligned} hyp_{4a} &= (\overline{Res\_vide} \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide} + Res\_plein \cdot Res\_vide) \cdot \overline{U_0} \\ &= (\overline{Res\_vide} \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide} + Res\_plein \cdot Res\_vide) \cdot \overline{Res\_plein \cdot Res\_vide} \\ &= \overline{Res\_vide} \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide} \end{aligned}$$

2) Nous continuons au point 2 du Q4-R3, en posant la question à l'utilisateur car $hyp_{4a} \neq 0*$. Nous demandons donc à l'utilisateur de compléter la spécification du système. C'est-à-dire, la question est de savoir si avec $\overline{Res\_vide} \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide}$ on met la pompe à un, à zéro ou si on la laisse dans l'état précédent :

ANALYSE PARTIELLE :

> Notons que, d'un coté, pour ce système faire le choix sur k_f équivaut à vouloir allumer la pompe dès que le niveau haut descend un peu ($\overline{Res\_plein}$). D'autre côté, faire le choix sur k_g équivaut à arrêter la pompe dès que le niveau dépasse un peu le niveau bas ($\overline{Res\_vide}$) même si la pompe venait de s'allumer après le signal du niveau bas.
>
> Il n'y a qu'une réponse possible de la part de l'utilisateur pour ce système :

☞ **REPONSE :** $\left\{ k_f = k_g = 0*, \quad k_h = hyp_v = \overline{Res\_vide} \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide} \right.$

C'est-à-dire, $A6)$ $\overline{Res\_vide} \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide} \leq REP_{Pomper}$

REMARQUE :

Cela veut dire que **dans l'incomplétude un choix sur k_f ou k_g ne doit pas être systématisé** car il existe le risque de provoquer un comportement non désiré.

Comme il est prévu pour la démarche manuelle de la résolution de la complétude, nous suivons la procédure [Q5-R2] au **P3** de l'algorithme, avant que la réponse soit considérée comme correcte et ajoutée au système au P4, car **il est nécessaire de faire la vérification de toutes les propriétés déjà calculées pour l'assertion qui vient d'être introduite** :

PROCÉDURE [Q5-R2] DE L'ALGORITHME 2 :

A) La sortie n'est pas bloquante (voir le système cohérent calculé au 7.1.2).

B) Nous passons à la vérification de la cohérence (point B du 6.3.1) avec l'expression hyp_1 :

$$=> \quad hyp_1 = \overline{U_0} \cdot \left((k_f \cdot k_g + k_f \cdot k_h + k_g \cdot k_h) + f \cdot (k_g + k_h) + g \cdot (k_f + k_h) + h \cdot (k_f \cdot k_g) \right)$$

Calculée avec l'information reçu par l'utilisateur dans la réponse précédent (k_f, k_g, et k_h). Nous avons donc après substitution :

$hyp_1 = 0*$.

C) Ne s'appliquent pas pour une seule sortie.

D) Ne s'appliquent pas pour des systèmes indépendants.

Nous avons donc en incluant l'assertion A6 introduite, au **P4** de l'algorithme 2, le système suivant :

$$\Rightarrow \begin{cases} Res\_vide \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide} \leq Pomper \\ (\overline{Res\_vide} \cdot Res\_plein + Bas\_vide \cdot \overline{Res\_plein}) \leq \overline{Pomper} \\ \overline{Res\_vide} \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide} \leq REP_{Pomper} \\ Res\_vide \cdot Res\_plein = 0* \end{cases}$$

D'après le résultat au **P1** : $hyp_4 = Res\_plein \cdot Res\_vide$, au deuxième cycle de calcul de l'algorithme, il est possible de considérer depuis la procédure [Q4-R3], que ce dernier système est complet parce que $hyp_4 = 0*$ car $Res\_plein \cdot Res\_vide = 0*$.

Étant donné que $hyp_4 = 0*$ **nous finissons l'algorithme 2**. Nous avons donc une solution séquentielle générale (voir [5.1]) : $SR(f + k_f, g + k_g)$. Ainsi pour notre exemple, nous avons :

$$Pomper = SR\left(\begin{matrix} Res\_vide \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide} + k_f, \\ \overline{Res\_vide} \cdot Res\_plein + Bas\_vide \cdot \overline{Res\_plein} + k_g \end{matrix} \right)$$

$$ssi \quad k_f + k_g \leq Res\_plein \cdot Res\_vide$$

Remarques :

Dans cette solution générale, la contrainte de cohérence entre les deux systèmes ($k_f \cdot k_g = 0*$) n'est pas considéré parce que $U_0 = Res\_plein \cdot Res\_vide = 0*$:

$$\left\{ k_f + k_g \leq 0* \right. \quad \Leftrightarrow \quad \begin{cases} k_f \leq 0* \\ k_g \leq 0* \end{cases} \quad \Leftrightarrow \quad \left\{ k_f \cdot k_g \leq 0* \right. \quad \Leftrightarrow \quad \left\{ k_f \cdot k_g = 0*. \right.$$

L'option de rester dans l'état précédent n'impose aucun changement d'état additionnel dans le comportement du système et en conséquence, en cas de panne des capteurs il se peut d'éviter sa propagation.

Il est possible donc d'utiliser le choix de rester dans l'état précédent : $hyp_v \le REP_{Pomper}$ ***pour une résolution systématique si*** $hyp_v \le U_0$.

Ainsi, nous avons pour cet exemple la solution suivante :

$k_f = 0*,\ k_g = 0*$ et $k_h = hyp_4 = Res\_plein \cdot Res\_vide$

$$\begin{cases} f = f + k_f \\ g = g + k_g \\ h = h + k_h \end{cases} \Rightarrow \begin{cases} Res\_vide \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide} \le Pomper \\ \overline{Res\_vide} \cdot Res\_plein + Bas\_vide \cdot \overline{Res\_plein} \le Pomper \\ \overline{Res\_vide} \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide} + Res\_plein \cdot Res\_vide \le REP_{Pomper} \end{cases}$$

Ce système calculé est cohérent est complètement spécifie. Il a comme solution SR[1] :

$$Pomper = SR\left(Res\_vide \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide},\ \overline{Res\_vide} \cdot Res\_plein + Bas\_vide \cdot \overline{Res\_plein} \right)$$

Nous précisons cependant que d'autres choix sont possibles tout en respectant $k_f + k_g \le Res\_plein \cdot Res\_vide$ dans la solution générale. Le désir de minimiser l'expression de la solution peut être la cause de la recherche d'autres choix dans la solution. En effet, le choix $hyp_v \le \overline{Pomper}$ a comme résultat une solution plus compacte :

Solution alternative) On peut choisir tout hyp_v mise sur k_g ($hyp_v \le \overline{Pomper}$) :

$k_f = 0*,\ k_g = hyp_v = Res\_plein \cdot Res\_vide$ et $k_h = 0*$

$$Pomper = SR\left(Res\_vide \cdot \overline{Res\_plein} \cdot \overline{Bas\_vide},\quad Res\_plein + Bas\_vide \right)$$

7.1.4 Implantation

Pour garantir qu'une solution existe pour les systèmes où les fonctions de sortie sont toutes indépendantes entre elles et que cette solution soit implantable, il ne faut que prouver que chaque sortie est correctement spécifiée parce que l'implantation n'apporte aucune contrainte additionnelle étant donné que les systèmes indépendants sont insensibles à l'ordre (6.3.4).

[1] Étant donné que nous avons $f \cdot g = 0*$ alors la solution $RS(f,g)$ est équivalente.

En effet, étant donné qu'il n'existe qu'une sortie l'implantation de la fonction solution peut être implantée sur un calculateur exécuté en parallèle et séquentiel; car l'ordre de calcul n'exerce aucune influence.

La figure 41 montre le diagramme d'implantation sur un calculateur exécuté en parallèle (dit câblé) et sur un calculateur exécuté en séquentiel. Les variables calculées par notre méthode nous servent pour l'implantation d'une façon transparente pour un automaticien en utilisant : $X_i = f_i + \overline{g_i} \cdot x_i$.

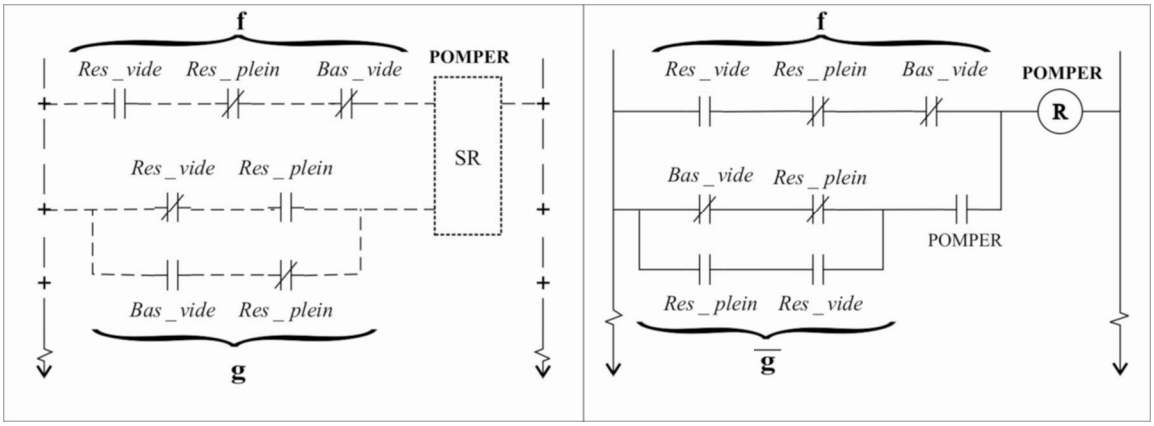


Figure 41 *Diagramme d'implantation de la fonction de sortie Pomper de l'exemple 1 : a) Diagramme Ladder et b) diagramme câblé*

7.1.5 Analyse du premier exemple

Ce cas nous permet de constater que l'algorithme proposé peut servir en premier lieu comme *un outil de vérification des propriétés* au cours de la synthèse et en cas d'erreur stopper le calcul au Q4 en choisissant R1. En seconde instance, il sert à la conception de la commande en détectant les oublis ou les manques de précisions sur la SF donnée par l'utilisateur. Une fois détectés, *les algorithmes indiquent les pas à suivre afin de corriger et surmonter le non accomplissement de ces propriétés*. Cette méthode permet une démarche évolutive où l'utilisateur donne de nouvelles informations afin d'enlever les contraintes. Pourtant, celles-ci n'obligent pas, dans l'algorithme 1, à refaire la vérification de tout le système, sinon uniquement de la partie ajoutée. D'autre part, l'exemple nous a permis d'introduire la démarche que nous suivrons pour introduire une méthodologie plus flexible, vis-à-vis de l'utilisateur, en utilisant la SF-T3. En effet, cette spécification élargit les possibilités de choix car elles sont toujours fausses. Ceci nous permettra de ne formuler des questions à l'utilisateur que si cela est indispensable, en utilisant les nouvelles options introduites (Q4-R3) dans l'algorithme 2.

7.2 Résolution de l'exemple 2 : Systèmes à deux sorties avec des contraintes sur les sorties

Le système étudié dans cette section a été formalisé dans la section 4.3.2. Le système d'équations formalisé en $\mathbb{I}$ est le suivant :

$$\begin{cases} A1) & TC \le Ouvrir \\ A2) & 3s/(PO \cdot \overline{TC} \cdot \overline{V}) \le Fermer \\ A3) & \overline{PF} \cdot V \le Ouvrir \\ A4) & Ouvrir \cdot Fermer = 0\,* \\ A5) & PO \le \overline{Ouvrir} \\ A6) & PF \le \overline{Fermer} \\ A7) & PO \cdot PF = 0\,* \end{cases}$$

L'objectif est donc d'exprimer les sorties *Ouvrir* et *Fermer* en utilisant un SR ou bien un RS, tout en respectant la spécification donnée. Les trois assertions *A1* à *A7*, ont été établies manuellement indépendamment les unes des autres. Si leur formalisation sur $\mathbb{I}$ permet de s'affranchir des imprécisions du LN, rien ne garantit que la réunion de ces expressions formelles constitue un système d'équations ayant une solution qui respecte les propriétés extrinsèques.

Dans cette section, nous décrirons la méthode de résolution des systèmes qui ont des sorties indépendantes tel que l'exemple précédent mais il est inclus une SF-T4. Nous devons suivre tous les pas de la méthode proposée car les simplifications de l'exemple précédent ne peuvent pas s'appliquer quand il y a une SF-T4.

Remarque :

Avant de continuer nous précisons qu'afin de simplifier les résultats partiels, nous allons utiliser les théorèmes des chapitres précédents. Cet exemple nous permet en plus de montrer l'utilité des théorèmes car ils sont utiles pour prouver des propriétés. Notamment nous soulignons dans cet exemple les théorèmes du TON. En effet, en utilisant l'assertion A7 et d'après A2; en utilisant [2.37] ⇔ [2.35] puis [3.43] et [2.47] nous avons,

$$\begin{array}{l} 3s/(PO \cdot \overline{TC} \cdot \overline{V}) \\ \\ 3s/(PO \cdot \overline{PF} \cdot \overline{TC} \cdot \overline{V}) \end{array} \quad \Rightarrow \quad \begin{cases} 3s/(PO \cdot \overline{TC} \cdot \overline{V}) \le PO \\ 3s/(PO \cdot \overline{TC} \cdot \overline{V}) \le \overline{PF} \\ 3s/(PO \cdot \overline{TC} \cdot \overline{V}) \le \overline{TC} \\ 3s/(PO \cdot \overline{TC} \cdot \overline{V}) \le \overline{V} \end{cases}$$

Depuis ces relations et en utilisant l'équivalence : [2.34] ⇔ [2.37] ⇔ [2.38] ⇔ [2.35] nous obtenons les relations suivantes que nous allons utiliser au fur et à mesure du besoin :

$$\left(\overline{PO}\cdot 3s/(PO\cdot\overline{TC}\cdot\overline{V})\right)=0* \quad \overline{PO}\cdot\overline{3s/\left(PO\cdot\overline{TC}\cdot\overline{V}\right)}=\overline{PO} \quad PO\cdot\left(3s/(PO\cdot\overline{TC}\cdot\overline{V})\right)=3s/(PO\cdot\overline{TC}\cdot\overline{V})$$
$$\left(PF\cdot 3s/(PO\cdot\overline{TC}\cdot\overline{V})\right)=0* \quad PF\cdot\overline{3s/\left(PO\cdot\overline{TC}\cdot\overline{V}\right)}=PF \quad \overline{PF}\cdot\left(3s/(PO\cdot\overline{TC}\cdot\overline{V})\right)=3s/(PO\cdot\overline{TC}\cdot\overline{V})$$
$$TC\cdot\left(3s/(PO\cdot\overline{TC}\cdot\overline{V})\right)=0* \quad TC\cdot\overline{3s/\left(PO\cdot\overline{TC}\cdot\overline{V}\right)}=TC \quad \overline{TC}\cdot\left(3s/(PO\cdot\overline{TC}\cdot\overline{V})\right)=3s/(PO\cdot\overline{TC}\cdot\overline{V})$$
$$V\cdot\left(3s/(PO\cdot\overline{TC}\cdot\overline{V})\right)=0* \quad V\cdot\overline{3s/\left(PO\cdot\overline{TC}\cdot\overline{V}\right)}=V \quad \overline{TC}\cdot\left(3s/(PO\cdot\overline{TC}\cdot\overline{V})\right)=3s/(PO\cdot\overline{TC}\cdot\overline{V})$$

7.2.1 Regroupement de la spécification de fonctionnement

D'après 6.3.3, en utilisant l'équation [2.46] nous allons réduire les assertions *A1*) et *A3*) et nous obtenons :

$$\begin{cases} A1\cdot A3) & TC+\overline{PF}\cdot V\le Ouvrir \\ A5) & PO\le\overline{Ouvrir} \\ A2) & 3s/(PO\cdot\overline{TC}\cdot\overline{V})\le Fermer \\ A6) & PF\le\overline{Fermer} \\ A4) & Ouvrir\cdot Fermer=0* \\ A7) & PO\cdot PF=0* \end{cases}$$

7.2.2 Vérification et corrections des propriétés

7.2.2.1 Vérification et résolution des incohérences du système de sorties

D'après 6.3.2, nous allons appliquer l'algorithme (1) pour la recherche et les corrections des incohérences sur les deux sorties au même temps en identifiant les variables pour chacune : Ouvrir et Fermer. Nous allons supposer que l'utilisateur veut résoudre toutes les incohérences sauf pour les cas des variables qui n'apparaissent jamais en absence de défaillance, c'est-à-dire U_0 (*A7*), car il n'y a aucune indication pour ces cas de figure. Cependant il demande que l'assertion *A4* ($Ouvrir\cdot Fermer=0*$) soit respectée même si une panne arrive.

Nous commençons le premier cycle de calcul de vérification par le calcul de la propriété de la cohérence : $f\cdot g+f\cdot h+g\cdot h=0*$. Ainsi, nous avons (à l'étape P1 du premier cycle de calcul de cet algorithme) pour les systèmes d'assertions $A1\cdot A3$ et $A5$ d'Ouvrir et $A2$ et $A6$ de Fermer :

P1_1) $hyp_{1O1}=\overline{\left(PO\cdot PF\right)}\cdot(f_O\cdot g_O+f_O\cdot h_O+g_O\cdot h_O)$
$=(\overline{PO}+\overline{PF})\cdot\left((TC+\overline{PF}\cdot V)\cdot(PO)+(TC)\cdot 0*+(\overline{PF}\cdot V)\cdot(0*)\right)$
$=PO\cdot\overline{PF}\cdot(TC+V)$

$$\begin{aligned} hyp_{1F1} &= \overline{\left(PO \cdot PF\right)} \cdot (f_F \cdot g_F + f_F \cdot h_F + g_F \cdot h_F) \\ &= (\overline{PO} + \overline{PF}) \cdot \left((3s/(PO \cdot \overline{TC} \cdot \overline{V}) \cdot (PF) + (3s/(PO \cdot \overline{TC} \cdot \overline{V}) \cdot 0* + (PF) \cdot (0*)\right) \\ &= 3s/(PO \cdot \overline{TC} \cdot \overline{V}) \cdot PF \cdot \overline{PO} \end{aligned}$$

L'hypothèse hyp_{1F1} est toujours fausse car, en utilisant le théorème [3.43] et [2.34] ⇔ [2.35] nous avons : $hyp_{1F1} = \left(3s/(PO \cdot \overline{TC} \cdot \overline{V}) \cdot (PO \cdot \overline{TC} \cdot \overline{V})\right) \cdot PF \cdot \overline{PO} = 0*$

Suivant l'algorithme et la disposition de l'utilisateur à résoudre lui-même les incohérences, nous ne lui posons la question que pour l'incohérence de la variable *Ouvrir* obtenue au P1_1 car la cohérence pour la variable *Fermer* est déjà vérifiée.

Q4_R2) $\left(PO \cdot \overline{PF} \cdot (TC + V)\right) \leq \{Ouvrir, \overline{Ouvrir}, REP_{Ouvrir}\}$?

> L'utilisateur peut s'apercevoir, à partir de la question, qu'il a oublié de bien préciser la SF car si la porte est déjà ouverte, il n'est pas nécessaire de commander la porte à ouvrir suit à la demande d'ouverture par la télécommande (A1) ou par la présence d'une la voiture (A3). De plus, la commande de fermeture prévoit dans le TON ($3s/(PO \cdot \overline{TC} \cdot \overline{V})$) que la porte ne fermera que 3 secondes après que la porte est ouvert et qu'il n'y a pas ni une voiture présente ni la télécommande active. Nous recevons donc comme réponse :

☞ **REPONSE :** $k_f = 0*,\ k_g = PO,\ k_h = 0*$

C'est-à-dire, $A8)\quad PO \leq \overline{Ouvrir}$

En continuant l'algorithme, une nouvelle itération de vérification de la cohérence pour la variable Ouvrir commence (P5) et nous obtenons pour la variable *Ouvrir* au P1 du deuxième cycle de calcul :

P1_2) $hyp_{1O2} = 0*$

L'incohérence enlevée, il ne nous reste qu'à finir l'algorithme pour obtenir les systèmes cohérents. Nous obtenons,

P6_1) $\begin{cases} PF \cdot TC + \overline{PF} \cdot \overline{PO} \cdot (TC + V) \leq Ouvrir \\ PO \leq \overline{Ouvrir} \end{cases}$ **P2_1)** $\begin{cases} 3s/(PO \cdot \overline{TC} \cdot \overline{V}) \leq Fermer \\ PF \leq \overline{Fermer} \end{cases}$

7.2.2.2 Vérification et corrections des incohérences dues aux états interdits

Une fois que le système de chaque variable de sortie (système de sorties) est cohérent, il faut vérifier les propriétés demandées sur le comportement global en commençant par le respect de l'état interdit (SF-T4) et en tout cas à corriger la commande s'il en a besoin.

Il faut donc vérifier la propriété de l'état interdit $Ouvrir \cdot Fermer$ qui ne doit jamais être commandé au même moment : $Ouvrir \cdot Fermer = 0*$. A partir des systèmes obtenus au **P6_1** et **P2_1**, (qui contiennent l'assertion A8 introduite par l'utilisateur), nous avons :

$$hyp_{fm} = \left(\frac{(f_{Fermer} + h_{Fermer} \cdot Fermer) \cdot (f_{Ouvrir} + h_{Ouvrir} \cdot Ouvrir)}{(f_{Fermer} + h_{Fermer} \cdot Fermer) \cdot (f_{Ouvrir} + h_{Ouvrir} \cdot)(Ouvrir \cdot Fermer)} \right) = 0*$$

Après substitution et réduction de l'expression en utilisant les expressions de la remarque 9, nous obtenons $hyp_{fm} = 0*$. **Cela veut dire que l'état** $Ouvrir \cdot Fermer$ **n'est pas accessible d'après cette spécification partielle car** $hyp_{fm} = 0*$. Nous soulignons cependant que ce résultat n'est pas encore suffisant pour garantir l'inaccessibilité de l'état car la complétude du système doit être vérifiée.

7.2.2.3 Vérification et correction de la SF afin que le système soit unité stable

Nous avons déterminé qu'une spécification partielle respecte l'unité stabilité si $hyp_{us} = 0*$ où :

$$hyp_{us} = \left(\prod_{i \in \{1,p\}} (f_{xi} + h_{xi} \cdot X_i) \right) \cdot \overline{fouh_{etat}} \text{ tel que } fouh_{etat} = \left(\prod_{i \in \{1,p\}} (f_{xi} + h_{xi} \cdot X_i) \right) \Big|_{\left(\prod_{i \in \{1,p\}} X_i \right) = 1*}$$

Calculons d'abord les $fouh_{etat}$ **des trois états non interdits** :

$$fouh_{Ouv \cdot \overline{Fer}} = (f_{Ouvrir} + h_{Ouvrir} \cdot Ouvrir) \cdot (g_{Fermer} + h_{Fermer} \cdot \overline{Fermer}) = PF \cdot TC$$

$$fouh_{\overline{Ouv} \cdot Fer} = \left((g_{Ouvrir} + h_{Ouvrir} \cdot \overline{Ouvrir}) \cdot (f_{Fermer} + h_{Fermer} \cdot Fermer) \right) = PO \cdot 3s / (PO \cdot \overline{TC} \cdot \overline{V})$$
$$= 3s / (PO \cdot \overline{TC} \cdot \overline{V})$$

$$fouh_{\overline{Ouv} \cdot \overline{Fer}} = \left((g_{Ouvrir} + h_{Ouvrir} \cdot \overline{Ouvrir}) \cdot (g_{Fermer} + h_{Fermer} \cdot \overline{Fermer}) \right) = PO \cdot PF = 0*$$

Ainsi, nous obtenons pour ces trois états :

$$hyp_{us(Ouv \cdot \overline{Fer})} = (PF \cdot TC) \cdot \overline{\left(PF \cdot TC \big|_{Ouvrir \cdot \overline{Fermer} = 1*} \right)} = (PF \cdot TC) \cdot \overline{PF \cdot TC} = 0*$$

$$hyp_{us(\overline{Ouv} \cdot Fer)} = \left(3s / (PO \cdot \overline{TC} \cdot \overline{V}) \right) \cdot \overline{\left(3s / (PO \cdot \overline{TC} \cdot \overline{V}) \big|_{\overline{Ouvrir} \cdot Fermer = 1*} \right)}$$
$$= (3s / (PO \cdot \overline{TC} \cdot \overline{V})) \cdot \overline{\left(3s / (PO \cdot \overline{TC} \cdot \overline{V}) \right)} = 0*$$

$$hyp_{us(\overline{Ouv} \cdot \overline{Fer})} = (0*) \cdot \overline{\left(0* \big|_{Ouvrir \cdot \overline{Fermer} = 1*} \right)} = (0*) \cdot \overline{0*} = 0*$$

Cela veut dire que pour tous les états non interdits, d'après la spécification (A1 à A8), **l'unité-stabilité est respectée**. Nous remarquons cependant que ce résultat n'est pas encore suffisant pour garantir l'unité-stabilité car il faut vérifier que le système est complet.

7.2.3 Résolution de l'incomplétude

Pour cet exemple, **nous allons choisir l'option de rester dans l'état précédent pour l'incomplétude**.

En suivant l'algorithme 2, dans un premier cycle de calcul, nous obtenons pour la vérification de l'incomplétude hyp_4 au **P1** les résultats suivants après simplification :

P1_1) $\begin{array}{l} hyp_{4ouvrir} = \overline{PO}\cdot\overline{TC}\cdot(PF+\overline{V}) \\ hyp_{4Fermer} = \overline{PF}\cdot\left(\overline{3s/(PO\cdot\overline{TC}\cdot\overline{V})}\right) \end{array}$

En considérant l'option prédéfinie de rester dans l'état précédent pour l'incomplétude :

☞ **PROPOSITION :** $k_{fouvrir}=0*,\ k_{gouvrir}=0*,\ k_{houvrir}=\overline{PO}\cdot\overline{TC}\cdot(PF+\overline{V}),$

$k_{ffermer}=0*,\ k_{gfermer}=0*$ et $k_{hfermer}=\overline{PF}\cdot\left(\overline{3s/(PO\cdot\overline{TC}\cdot\overline{V})}\right)$

C'est-à-dire, $\begin{cases} A9) & \overline{PO}\cdot\overline{TC}\cdot(PF+\overline{V}) \le REP_{Ouvrir} \\ A10) & \overline{PF}\cdot\left(\overline{3s/(PO\cdot\overline{TC}\cdot\overline{V})}\right) \le REP_{Fermer} \end{cases}$

En considérant ces assertions nous avons le système suivant au P4 de l'algorithme 2 :

P4_1) $\begin{cases} PF\cdot TC+\overline{PF}\cdot\overline{PO}\cdot(TC+V) \le Ouvrir \\ PO \le \overline{Ouvrir} \\ \overline{PO}\cdot\overline{TC}\cdot(PF+\overline{V}) \le REP_{Ouvrir} \end{cases}$ **P4_1)** $\begin{cases} 3s/(PO\cdot\overline{TC}\cdot\overline{V}) \le Fermer \\ PF \le \overline{Fermer} \\ \overline{PF}\cdot\left(\overline{3s/(PO\cdot\overline{TC}\cdot\overline{V})}\right) \le REP_{Fermer} \end{cases}$

En continuant avec l'algorithme, nous suivons l'option Q5-R2 :

PROCÉDURE [Q5-R2] DE L'ALGORITHME 2 :

A) **NON BLOCAGE.** Il s'avère, depuis les systèmes montrés ci-dessus, que les sorties ne sont pas bloquantes.

B) **COHÉRENCE.** Comme toute l'incomplétude, ce qui n'était pas choisi, est mis en mémoire alors les systèmes restent cohérents.

C) **ÉTATS INTERDITS.** Nous pouvons vérifier le respect des états interdits car les systèmes des sorties sont cohérents et ces systèmes ont été complétés. Nous calculons le respect des états interdits, tel que nous l'avons fait dans la section précédente mais pour les systèmes complets :

$$hyp_{fm} = \left(\begin{array}{r} (F_{Fermer}+H_{Fermer}\cdot Fermer)\cdot(F_{Ouvrir}+H_{Ouvrir}\cdot Ouvrir) \\ (F_{Fermer}+H_{Fermer}\cdot Fermer)\cdot(F_{Ouvrir}+H_{Ouvrir}\cdot)(Ouvrir\cdot Fermer) \end{array} \right)$$

En substituant les valeurs du système complété (voir **P4_1**), il vient

$$hyp_{fm} = (TC + V) \cdot \overline{PO} \cdot \overline{PF} \cdot (\overline{Ouvrir} \cdot Fermer)$$

Comme $hyp_{fm} \neq 0*$ et la SF-T4 interdisant l'état doit être respectée alors l'utilisateur doit proposer de corrections en choisissant un état différent de tel sort que cet état ne soit pas l'état interdit ($Ouvrir \cdot Fermer$). Cela veut dire, que l'utilisateur doit valider l'ajout d'une assertion afin que l'état ne soit pas accessible $Ouvrir \cdot Fermer = 0*$.

ANALYSE PARTIELLE :

Nous allons analyser le premier terme de hyp_{fm}. Il s'avère que l'utilisateur a demandé dans la spécification d'ouvrir le portail, si la télécommande est commandée (assertion A1) et si une voiture est présente tant que le portail n'est pas fermé (A3). Ainsi, l'ouverture serait demandée si la télécommande s'active ou une voiture est présente quand le portail est en train de se fermer par contre la fermeture ne s'arrêtera pas et par conséquence l'état dit interdit deviendrait accessible. De ce fait, l'utilisateur doit choisir un autre état d'arrivée différent de $Ouvrir \cdot Fermer$ et de $\overline{Ouvrir} \cdot Fermer$:

$$(TC + V) \cdot \overline{PO} \cdot \overline{PF} \cdot (\overline{Ouvrir} \cdot Fermer) \leq \overline{\left(Ouvrir \cdot Fermer\right)} \cdot \overline{\left(\overline{Ouvrir} \cdot Fermer\right)}$$
$$(TC + V) \cdot \overline{PO} \cdot \overline{PF} \cdot (\overline{Ouvrir} \cdot Fermer) \leq \overline{Fermer}$$

Nous remarquons qu'en principe l'utilisateur n'a pas que ce choix comme réponse. Il s'avère que la fermeture doit s'arrêter toujours si la télécommande est activée ou il y a une voiture présente. C'est-à-dire, l'arrêt de la fermeture ne dépend pas de l'état $\overline{Ouvrir} \cdot Fermer$. Ainsi, l'utilisateur peut donner une réponse plus ciblée en introduisant au système l'assertion :

☞ **REPONSE :** $\left\{A8\right)\quad (TC + V) \cdot \overline{PO} \cdot \overline{PF} \leq \overline{Fermer}$

Il faut vérifier cependant que la cohérence est respectée avec la nouvelle assertion. Nous avons donc,

$k_{fouvrir} = k_{gouvrir} = k_{ffermer} = 0*$, $\quad k_{gfermer} = (TC + V) \cdot \overline{PO} \cdot \overline{PF}$,

$k_{houvrir} = \overline{f_{ouvrir}} \cdot \overline{g_{ouvrir}} \cdot \overline{h_{ouvrir}}$, $\quad k_{hfermer} = \overline{f_{fermer}} \cdot \overline{g_{fermer}} \cdot \overline{h_{fermer}} \cdot \overline{\left((TC + V) \cdot \overline{PO} \cdot \overline{PF}\right)}$,

et comme résultat

$hyp_{1ouvrir} = 0*$ et $hyp_{1fermer} = 0*$.

D) Il faut vérifier l'unité-stabilité car il y a eu une assertion qui a été ajouté (A8) au point 2. En substituant donc les nouvelles valeurs nous avons

$$FouH_{Ouv\cdot\overline{Fer}} = PF\cdot TC + \overline{PO}\cdot(PF\cdot Ouvrir + TC + V\cdot\overline{PF} + Ouvrir\cdot\overline{Fermer})$$
$$FouH_{\overline{Ouv}\cdot Fer} = \overline{PF}\cdot(PO + \overline{TC}\cdot\overline{V}\cdot\overline{Ouvrir})\cdot Fermer + 3s/(PO\cdot\overline{TC}\cdot\overline{V})$$
$$Fouh_{\overline{Ouv}\cdot\overline{Fer}} = PF\cdot(PO + \overline{TC}\cdot\overline{Ouvrir}) + \overline{\left(3s/(PO\cdot\overline{TC}\cdot\overline{V})\right)}\cdot(PO + \overline{TC}\cdot\overline{V}\cdot\overline{Ouvrir})\cdot\overline{Fermer}$$

Le résultat du calcul des conditions d'existence de la solution respectant l'unité-stabilité est :

$$hyp_{us(Ouv\cdot\overline{Fer})} = FouH_{Ouv\cdot\overline{Fer}}\cdot\overline{\left(FouH_{Ouv\cdot\overline{Fer}}\big|_{Ouv=1^*,Fer=0^*}\right)} = 0^*$$
$$hyp_{us(\overline{Ouv}\cdot Fer)} = FouH_{\overline{Ouv}\cdot Fer}\cdot\overline{\left(FouH_{\overline{Ouv}\cdot Fer}\big|_{Ouv=0^*,Fer=1^*}\right)} = 0^{**}$$
$$hyp_{us(\overline{Ouv}\cdot\overline{Fer})} = FouH_{\overline{Ouv}\cdot\overline{Fer}}\cdot\overline{\left(FouH_{\overline{Ouv}\cdot\overline{Fer}}\big|_{Ouv=0^*,Fer=0^*}\right)} = 0^*$$

Cela veut dire que le système, en incluant l'assertion A8, respect toutes les propriétés en considérant que l'incomplétude reste dans l'état précédent (solution séquentielle donnée par [5.2]). Nous présentons ci-dessous le système et les fonctions des sorties à la sortie du calculateur en utilisant un RS pour *Ouvrir* et un SR pour *Fermer* .

$$\begin{cases} PF\cdot TC + \overline{PF}\cdot\overline{PO}\cdot(TC+V) \le Ouvrir \\ PO \le \overline{Ouvrir} \\ \overline{PO}\cdot\overline{TC}\cdot(PF+\overline{V}) \le REP_{Ouvrir} \end{cases} \quad \begin{cases} 3s/(PO\cdot\overline{TC}\cdot\overline{V}) \le Fermer \\ PF + (TC+V)\cdot\overline{PO} \le \overline{Fermer} \\ \overline{PF}\cdot(PO + \overline{TC}\cdot\overline{V} + Ouvrir)\cdot\overline{\left(3s/(PO\cdot\overline{TC}\cdot\overline{V})\right)} \le REP_{Fermer} \end{cases}$$

$$\begin{cases} SF-T3) & PO\cdot PF = 0^* \\ SF-T4) & Ouvrir\cdot Fermer = 0^* \end{cases}$$

$$Ouvrir = SR(f_{Ov}, g_{Ov}) = SR\left(PF\cdot TC + \overline{PF}\cdot\overline{PO}\cdot(TC+V), PO\right)$$
$$Fermer = SR(f_{Ov}, g_{Ov}) = SR\left(3s/(PO\cdot\overline{TC}\cdot\overline{V}), PF + (TC+V)\cdot\overline{PO}\right)$$

Nous passerons à l'implantation mais nous remarquons que *Ouvrir* peut être ramené vers une expression plus courte en utilisant des théorèmes de SR. En effet, en utilisant la solution RS du [5.2],[3.28],[2.3],[2.10],[2.3],[3.28],[2.3],[2.3],[2.8] et [2.5] nous simplifions la solution :

$$\begin{aligned} Ouvrir &= RS\left(\left(PF\cdot TC + \overline{PF}\cdot\overline{PO}\cdot(TC+V)\right), PO\right) = RS\left(\left(PF\cdot TC + \overline{PF}\cdot\overline{PO}\cdot(TC+V)\right)\cdot\overline{PO}, PO\right) \\ &= RS\left(\left(PF\cdot TC + \overline{PF}\cdot(TC+V)\right)\cdot\overline{PO}, PO\right) = RS\left(PF\cdot TC + \overline{PF}\cdot(TC+V), PO\right) \\ &= RS\left(TC + \overline{PF}\cdot V, PO\right) \end{aligned}$$

7.2.4 Implantation

L'implantation sur un calculateur demande en plus la semi-insensibilité à l'ordre. La figure 42 montre le graphe de dépendance. Nous remarquons que les sorties *Ouvrir* et *Fermer* sont indépendantes car aucune des deux à une affectation sur l'autre. En conséquence

l'ordonnancement des sorties *Ouvrir* et *Fermer* n'a aucune influence sur le système car il est insensible à l'ordre.

D'autre part, la variable interne temporelle TON nommée $3s/(PO \cdot \overline{TC} \cdot \overline{V})$, ne dépend que des entrées et elle affecte *Fermer*, par conséquence elle doit être calculée avant *Fermer* afin de ne pas provoquer la non unité stabilité.

Figure 42 *Graphe de dépendances*

La figure 43a montre le diagramme d'implantation sur un calculateur exécuté en séquentiel avec le même ordre qui respecte la semi-insensibilité à l'ordre.

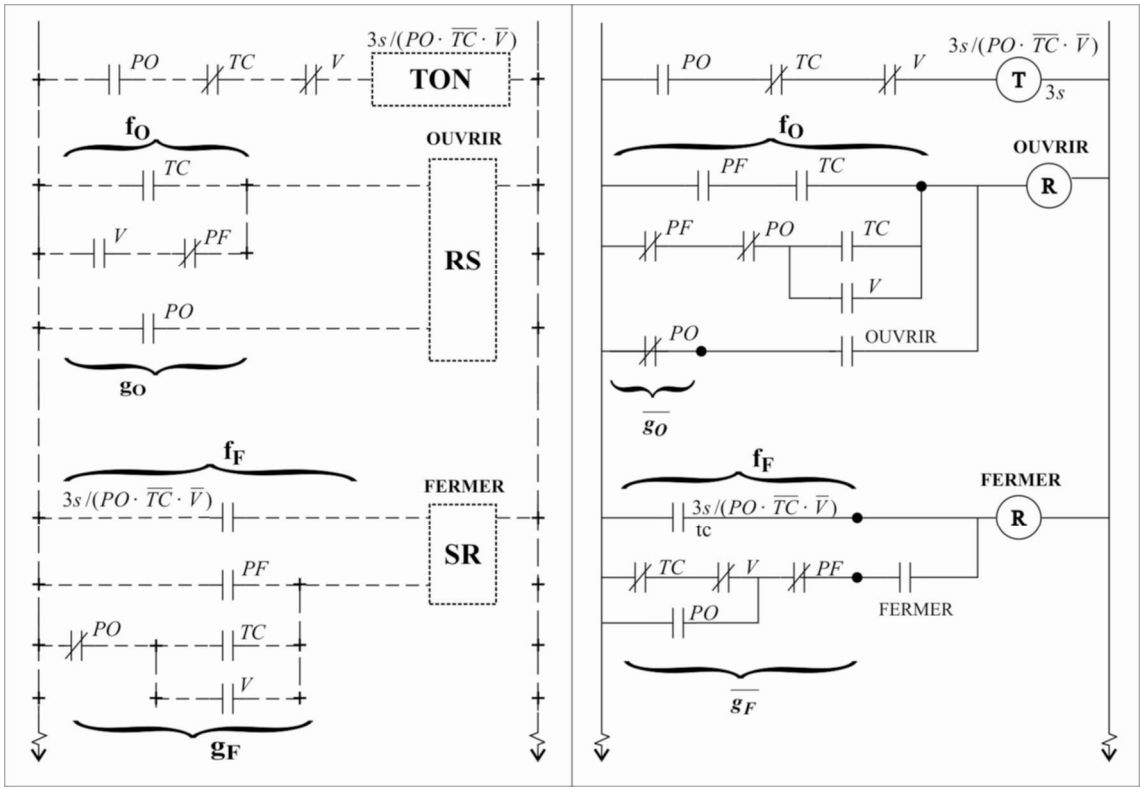


Figure 43 *Diagramme d'implantation des fonctions de sortie Ouvrir et Fermer de l'exemple 2*
a) Diagramme Ladder et b) diagramme câblé

D'autre part, la figure 43b montre l'implantation sur un calculateur exécuté en parallèle (dit câblé). Les variables calculées par notre méthode nous servent pour l'implantation d'une façon transparente pour un automaticien en utilisant : $X_i = f_i + \overline{g_i} \cdot x_i$.

Pour le cas de l'exécution en parallèle nous pouvons utiliser des équivalences pour simplifier la solution car les variables *f*, *g* et *h* sont disjointes et complémentaires. En effet, nous avons

$Ouvrir = f_O + h_O \cdot Ouvrir = f_O + (f_O + h_O) \cdot Ouvrir = f_O + (\overline{g_O}) \cdot Ouvrir$, et

$Fermer = f_F + h_F \cdot Fermer = f_F + (f_F + h_F) \cdot Fermer = f_F + (\overline{g_F}) \cdot Fermer$.

7.2.5 Analyse du deuxième exemple

Ce cas nous a permis de montrer qu'une fois calculée l'expression définissant la vérification d'une propriété, il est possible de faire des vérifications même du comportement global du système, tel qu'un état interdit, sans faire la composition de la SF définie pour chaque sortie. Une vérification partielle est possible dans certains cas même si le système n'est pas complet.

Cet exemple nous a permis, en plus, de montrer une démarche systématique de résolution, une fois que les contraintes données, dans la SF de l'utilisateur, sont surmontées. Pour arriver à ce but, nous avons montré que parfois il faut ajouter des assertions pour mettre à 1 ou à 0 un signal de sortie pour faire respecter la SF-T4. Autrement dit, le fait de choisir toute l'incomplétude à rester dans l'état précédent sans vérifier les propriétés n'est pas recommandé.

De plus, nous avons fait appel à des théorèmes sur l'algèbre $\mathbb{I}$ de l'opérateur TON pour vérifier des propriétés tel que la cohérence. Nous remarquons que ces vérifications sont possibles car ces opérateurs ont été définis par des expressions temporelles dans l'algèbre $\mathbb{I}$ et, en utilisant les théorèmes développés précédemment, la vérification reste dans le plan combinatoire. Or, il n'est plus nécessaire de refaire le calcul par des expressions temporelles.

7.3 Conclusions

Nous avons utilisé le premier exemple, qui ne considère que des SF-T1, pour bien illustrer tous les pas des algorithmes. De plus, nous montrons l'effet de ne pas considérer des SF-T3 dans la SF et que le choix dans l'incomplétude doit prendre en compte l'interaction avec la partie opérative. Une famille de solutions a été proposée d'abord afin de mettre en évidence l'importance de considérer le cas général du chapitre 5 et ensuite, une solution a été choisie afin de présenter l'implantation.

Dans le deuxième exemple l'opération TON nous a permis d'illustrer l'utilisation des opérations temporelles et ses théorèmes. Une SF-T4 a été utilisée pour montrer la démarche de vérifications et corrections d'une spécification de ce type. La démarche systématique proposée pour résoudre l'incomplétude a été mise en évidence. Nous avons profité pour introduire l'utilisation des graphes de dépendances pour la vérification de la semi-insensibilité à l'ordre.

Chapitre 8

Dans ce chapitre nous résolvons l'exemple 3 formalisé au chapitre 4. Cet exemple utilise les trois algorithmes de vérification/correction des propriétés. L'intervention de l'utilisateur est mise en valeur afin de résoudre les incohérences ou le manque d'informations dans la SF pour pouvoir calculer la fonction de contrôle.

A travers cet exemple nous montrons en plus, que le choix de prendre en compte une option prédéfinie pour la résolution de l'incomplétude permet d'automatiser la résolution de l'incomplétude sans la participation de l'utilisateur. La génération automatique des fonctions de contrôle par la méthode proposée ne dépendra que de la définition sans incohérence de la SF.

8. Étude de cas : Exemple 3

Nous présentons dans ce chapitre le troisième exemple qui concerne la commande d'un système à plusieurs sorties avec dépendances : la commande d'un préhenseur pneumatique.
Nous rappelons que la méthode commence à partir de la spécification formelle, en algèbre $\mathbb{I}$, de la commande et du système physique.

Avant de résoudre cet exemple, qui a été formalisé dans le chapitre 4, nous devons faire une précision sur la formalisation de l'assertion 7 :

Remarque :

La spécification en langage naturelle de l'assertion A7 :

A7) En étant toujours au poste de dépose, une fois en position haute, **reculer le préhenseur JUSQU'AU** point de départ (au poste de prise et en position haute).

Nous avons représenté les contraintes *de mise à un* de la commande de reculer par l'assertion A7a et *la mise à zéro* de reculer par l'assertion A7b.
Il est vrai cependant que nous n'avons pas représenté l'action demandée de continuer à reculer jusqu'à l'arrivée au point de départ.

L'introduction de l'opération REP au chapitre 5 nous permet maintenant de compléter cette spécification par l'assertion :

$$\left\{ A7c) \quad \overline{\left(P\_depose \cdot P\_haute\right)} \cdot \overline{\left(P\_prise \cdot P\_haute\right)} \leq REP_{Reculer} \right.$$

Ainsi, l'assertion A7a indique la contrainte de Reculer quand $P\_depose \cdot P\_haute$ est vrai, l'assertion A7c demande de continuer à reculer et l'assertion A7b demande d'arrêter de reculer.

Nous allons suivre, dans la section suivante, la méthode décrite au chapitre 6 pour résoudre ce système d'équations en incluant l'assertion A7c.

8.1 **Résolution de l'exemple 3 : Système à plusieurs sorties**

Le système étudié dans cette section a été formalisé dans la section 4.3.3. Le système d'équations formalisé en $\mathbb{I}$, en incluant A7c, est le suivant :

Spécification du cycle en U :

$$\begin{cases} A1) & \left(P\_prise \cdot P\_haute \cdot \uparrow dcy\right) \le Descendre \\ A2) & P\_prise \cdot P\_basse \le Aspirer \\ A3) & 1s/(P\_prise \cdot P\_basse) \le \overline{Descendre} \\ A4) & P\_prise \cdot P\_haute \le Avancer \\ A5) & P\_depose \cdot P\_haute \le \overline{Avancer} \cdot Descendre \\ A6) & P\_depose \cdot P\_basse \le \overline{Descendre} \cdot \overline{Aspirer} \\ A7a) & P\_depose \cdot P\_haute \le Reculer \\ A7b) & P\_prise \cdot P\_haute \le \overline{Reculer} \\ A7c) & \overline{\left(P\_depose \cdot P\_haute\right)} \cdot \overline{\left(P\_prise \cdot P\_haute\right)} \le REP_{Reculer} \\ A8) & ARU \le \overline{Avancer} \cdot \overline{Reculer} \cdot \overline{Descendre} \end{cases}$$

Spécifications dues aux contraintes sur les entrées :

$$\begin{cases} A9) & P\_prise \cdot P\_depose = 0^* \\ A10) & P\_haute \cdot P\_basse = 0^* \end{cases}$$

Spécifications de sécurité :

$$\begin{cases} A11) & \overline{P\_haute} \cdot Avancer = 0^* \\ A12) & \overline{P\_haute} \cdot Reculer = 0^* \\ A13) & \overline{P\_prise} \cdot \overline{P\_depose} \cdot Descendre = 0^* \\ A14) & Avancer \cdot Reculer = 0^* \\ A15) & Avancer \cdot Descendre = 0^* \\ A16) & Reculer \cdot Descendre = 0^* \end{cases}$$

L'objectif est donc d'exprimer les sorties : $Avancer$, $Reculer$, $Descendre$ et $Aspirer$ en utilisant un SR ou bien un RS, tout en respectant la spécification donnée. Les 16 assertions *A1 à A16*, ont été établies manuellement indépendamment les unes des autres. Si leur formalisation sur $\mathbb{I}$ permet de s'affranchir des imprécisions du LN, rien ne garantit que la réunion de ces expressions formelles constitue un système d'équations ayant une solution qui respecte en plus les propriétés extrinsèques.

Remarques :

Avant de continuer nous précisons qu'afin de simplifier les résultats partiels, nous allons utiliser les théorèmes des chapitres précédents. Cet exemple nous permet en plus de montrer l'utilité des théorèmes car ils sont indispensables pour prouver les propriétés. Notamment nous soulignons dans cet exemple les théorèmes du TON. En effet, en utilisant les assertions A9 et A10 et d'après A3; en utilisant [2.37] ⇔ [2.35] et ensuite [3.43] et [2.47] nous avons,

$$\begin{array}{c} 1s/(P\_prise \cdot P\_basse) \\ 1s/(P\_prise \cdot P\_basse \cdot \overline{P\_depose} \cdot \overline{P\_haute}) \end{array} \Rightarrow \begin{cases} 1s/(P\_prise \cdot P\_basse) \le P\_prise \\ 1s/(P\_prise \cdot P\_basse) \le P\_basse \\ 1s/(P\_prise \cdot P\_basse) \le \overline{P\_depose} \\ 1s/(P\_prise \cdot P\_basse) \le \overline{P\_haute} \end{cases}$$

Depuis ces relations et en utilisant l'équivalence : [2.34] ⇔ [2.37] ⇔ [2.38] ⇔ [2.35] nous obtenons les relations suivantes que nous allons utiliser au fur et à mesure du besoin :

$$\overline{P\_prise} \cdot \left(1s/(P\_prise \cdot P\_basse)\right) = 0* \qquad P\_haute \cdot \left(1s/(P\_prise \cdot P\_basse)\right) = 0*$$
$$\overline{P\_basse} \cdot \left(1s/(P\_prise \cdot P\_basse)\right) = 0* \qquad P\_depose \cdot \left(1s/(P\_prise \cdot P\_basse)\right) = 0*$$
$$\overline{P\_prise} \cdot \overline{1s/(P\_prise \cdot P\_basse)} = \overline{P\_prise} \qquad P\_haute \cdot \overline{1s/(P\_prise \cdot P\_basse)} = P\_haute$$
$$\overline{P\_basse} \cdot \overline{1s/(P\_prise \cdot P\_basse)} = \overline{P\_basse} \qquad P\_depose \cdot \overline{1s/(P\_prise \cdot P\_basse)} = P\_depose$$

$$P\_prise \cdot \left(1s/(P\_prise \cdot P\_basse)\right) = 1s/(P\_prise \cdot P\_basse)$$
$$P\_basse \cdot \left(1s/(P\_prise \cdot P\_basse)\right) = 1s/(P\_prise \cdot P\_basse)$$
$$\overline{P\_haute} \cdot \left(1s/(P\_prise \cdot P\_basse)\right) = 1s/(P\_prise \cdot P\_basse)$$
$$\overline{P\_depose} \cdot \left(1s/(P\_prise \cdot P\_basse)\right) = 1s/(P\_prise \cdot P\_basse)$$

8.1.1 Regroupement des spécifications

1) Nous exprimons les assertions qui représentent les propriétés à respecter *A11*, *A12* et *A13* comme relations d'ordre ([2.37] ⇔ [2.34]).

$$\begin{cases} A11) & \overline{P\_haute} \le \overline{Avancer} \\ A12) & \overline{P\_haute} \le \overline{\text{Re}culer} \\ A13) & \overline{P\_prise} \cdot \overline{P\_depose} \le \overline{Descendre} \end{cases}$$

2) Les assertions *A1*, *A2*, *A3*, *A4*, *A7a*, *A7b*, *A1*, *A12* et *A13* sont des **SF-T1**. L'assertion *A7c* est une **SF-T5**. Les assertions *A5*, *A6* et *A8* sont des **SF-T2**. Les assertions *A14*, *A15* et *A16* sont des **SF-T4**. Enfin, les assertions *A9* et *A10* sont des **SF-T3**.

3) Nous décomposons les assertions *A5*, *A6* et *A8* car elles ont un état d'arrivée ([2.47]).

$$\{A5) \quad P\_depose \cdot P\_haute \le \overline{Avancer} \cdot Descendre \quad \Leftrightarrow \quad \begin{cases} A5a) & P\_depose \cdot P\_haute \le \overline{Avancer} \\ A5b) & P\_depose \cdot P\_haute \le Descendre \end{cases}$$

$$\{A6) \quad P\_depose \cdot P\_basse \le \overline{Descendre} \cdot \overline{Aspirer} \quad \Leftrightarrow \quad \begin{cases} A6a) & P\_depose \cdot P\_basse \le \overline{Descendre} \\ A6b) & P\_depose \cdot P\_basse \le \overline{Aspirer} \end{cases}$$

$$\{A8) \quad ARU \le \overline{Avancer} \cdot \overline{Reculer} \cdot \overline{Descendre} \quad \Leftrightarrow \quad \begin{cases} A8a) & ARU \le \overline{Avancer} \\ A8b) & ARU \le \overline{Reculer} \\ A8c) & ARU \le \overline{Descendre} \end{cases}$$

4, 5 et 6) Le point 5 ne s'applique pas car il n'y a plus de SF-T2. Nous regroupons donc toutes les assertions du point 3 car elles ont toutes comme arrivée une sortie, avec les SF-T1. Nous

soulignons que les propriétés sont regroupées comme prioritaires (A11 à A16). Le regroupement est fait par sortie en utilisant [2.46] en considérant aussi l'assertion *A7c* (SF-T5) car elle est définie pour une sortie, comme suit :

$$\begin{cases} \begin{cases} A4) & P\_prise \cdot P\_haute \le Avancer \\ A5a \cdot A8a \cdot A11) & P\_depose \cdot P\_haute + ARU \le \overline{Avancer} \end{cases} \\ \begin{cases} A7a) & P\_depose \cdot P\_haute \le \mathrm{Re}culer \\ A7b \cdot A8b \cdot A12) & P\_prise \cdot P\_haute + ARU \le \overline{\mathrm{Re}culer} \\ A7c) & \overline{P\_haute} + \overline{P\_depose} \cdot \overline{P\_prise} \le REP_{Reculer} \end{cases} \\ \begin{cases} A1 \cdot A5b) & \left(P\_prise \cdot P\_haute \cdot \uparrow dcy + P\_depose \cdot P\_haute\right) \le Descendre \\ A3 \cdot A6a \cdot A8c \cdot A13) & \left(1s/(P\_prise \cdot P\_basse) + P\_depose \cdot P\_basse + ARU\right) \le \overline{Descendre} \end{cases} \\ \begin{cases} A2) & P\_prise \cdot P\_basse \le Aspirer \\ A6b) & P\_depose \cdot P\_basse \le \overline{Aspirer} \end{cases} \end{cases}$$

En incluant les propriétés à respecter comme prioritaires (A11 à A16) :

$$\begin{cases} \begin{cases} A4) & P\_prise \cdot P\_haute \le Avancer \\ A5a \cdot A8a \cdot P1) & P\_depose + \overline{P\_haute} + ARU \le \overline{Avancer} \end{cases} \\ \begin{cases} A7a) & P\_depose \cdot P\_haute \le \mathrm{Re}culer \\ A7b \cdot A8b \cdot P2) & P\_prise + \overline{P\_haute} + ARU \le \overline{\mathrm{Re}culer} \\ A7c) & \overline{P\_haute} + \overline{P\_depose} \cdot \overline{P\_prise} \le REP_{Reculer} \end{cases} \\ \begin{cases} A1 \cdot A5b) & \left(P\_prise \cdot P\_haute \cdot \uparrow dcy + P\_depose \cdot P\_haute\right) \le Descendre \\ A3 \cdot A6a \cdot A8c \cdot P3) & \left(\begin{matrix} 1s/(P\_prise \cdot P\_basse) + P\_depose \cdot P\_basse \\ + ARU + \overline{P\_prise} \cdot \overline{P\_depose} \end{matrix}\right) \le \overline{Descendre} \end{cases} \\ \begin{cases} A2) & P\_prise \cdot P\_basse \le Aspirer \\ A6b) & P\_depose \cdot P\_basse \le \overline{Aspirer} \end{cases} \end{cases}$$

7) Regroupons les SF-T3 et SF-T4 en utilisant [2.25]

$$\begin{cases} A14 \cdot A15 \cdot A16) & Avancer \cdot \mathrm{Re}culer + Avancer \cdot Descendre + \mathrm{Re}culer \cdot Descendre = 0* \\ A9 \cdot A10) & P\_prise \cdot P\_depose + P\_haute \cdot P\_basse = 0* \end{cases}$$

Après le regroupement nous constatons qu'il a le système d'équations des sorties (points 4, 5 et 6). En plus, d'après le point 7 nous avons une équation (SF-T4) qui représente tous les états interdits et une équation (SF-T3) qui représente tous les hypothèses sur les entrées. D'autre part, nous constatons que pour cet exemple nous n'avons plus des équations du système de comportement global car toutes les assertions SF-T2 ont été décomposées en SF-T1.

8.1.2 Vérification et correction des propriétés

8.1.2.1 Détection et résolution des incohérences des sorties

Nous allons appliquer l'algorithme 1 pour la recherche et les corrections des incohérences sur les sorties au même temps en identifiant les variables pour chacune : *Avancer*, *Reculer*, *Descendre* et *Aspirer*. Nous ferons appel à l'utilisateur pour résoudre les incohérences sauf pour les cas des variables qui n'apparaissent jamais en absence de défaillance, c'est-à-dire U0. Cependant il est demandé que les assertions *A14, A15* et *A16* soient respectées même si une panne arrive ($Avancer \cdot Reculer + Avancer \cdot Descendre + Reculer \cdot Descendre = 0*$).

Nous commençons la première vérification par le calcul de la propriété de la cohérence : hyp_1

P1_1) $hyp_{1Av} = Phaute \cdot Pprise \cdot (ARU + Pdepose)$

$$hyp_{1Rec} = Phaute \cdot Pprise \cdot Pdepose + ARU \cdot Phaute \cdot (Pdepose + \overline{Pprise})$$

$$hyp_{1Des} = Phaute \cdot \begin{pmatrix} Pdepose \cdot P\_basse + ARU \cdot (\uparrow dcy \cdot Pprise + Pdepose) \\ +1s/(P\_prise \cdot P\_basse) \cdot (\uparrow dcy \cdot Pprise + Pdepose) \end{pmatrix}$$

$$hyp_{1Asp} = P\_basse \cdot Pdepose \cdot Pprise$$

Remarque :

En utilisant des théorèmes il est possible dans certains cas de réduire ces expressions. En effet, en utilisant les expressions au début de ce chapitre nous pouvons simplifier hyp_{1Des}. En effet, $P\_haute \cdot (1s/(P\_prise \cdot P\_basse)) = 0*$ et **par conséquence** hyp_{1Des} **peut être simplifiée comme :**

$$\boldsymbol{hyp_{1Des} = Phaute \cdot (Pbasse \cdot Pdepose + ARU \cdot (\uparrow dcy \cdot Pprise + Pdepose))}$$

Nous continuons avec l'algorithme 1 et on s'arrête au **Q4-R2** pour lui poser les questions suivantes sur les systèmes incohérents. Notons que *Aspirer* est déjà cohérent

Q4_R2)

$$ARU \cdot (Phaute \cdot Pprise \cdot \overline{Pbasse} \cdot \overline{Pdepose}) \leq \{Avancer, \overline{Avancer}, REP_{Avancer}\} ?$$

$$ARU \cdot (Phaute \cdot \overline{Pbasse} \cdot \overline{Pprise}) \leq \{Reculer, \overline{Reculer}, REP_{Reculer}\} ?$$

$$ARU \cdot Phaute \cdot \overline{Pbasse} \cdot (\uparrow dcy \cdot Pprise \cdot \overline{Pdepose} + Pdepose \cdot \overline{Pprise}) \leq \{Descendre, \overline{Descendre}, REP_{Descendre}\} ?$$

ANALYSE PARTIELLE :

Notons que les expressions calculées ont toutes comme facteur l'arrête d'urgence (ARU). Nous allons considérer que le signal ARU doit impérativement provoquer l'arrêt de la commande des sorties *Avancer*, *Reculer* et *Descendre*. Autrement dit, l'arrêt est prioritaire.

Nous continuons l'algorithme et nous recevons évidement comme réponse :

$$k_{gAvancer} \Rightarrow ARU \cdot (Phaute \cdot Pprise \cdot \overline{Pbasse} \cdot \overline{Pdepose}) \leq \overline{Avancer}$$

$$k_{gReculer} \Rightarrow ARU \cdot (Phaute \cdot \overline{Pbasse} \cdot \overline{Pprise}) \leq \overline{Reculer}$$

$$k_{gDescendre} \Rightarrow ARU \cdot Phaute \cdot \overline{Pbasse} \cdot \begin{pmatrix} \uparrow dcy \cdot Pprise \cdot \overline{Pdepose} \\ +Pdepose \cdot \overline{Pprise} \end{pmatrix} \leq \overline{Descendre}$$

Ainsi, en suivant l'algorithme 1, nous calculons hyp_1 à nouveau, au deuxième cycle et nous obtenons :

P1_1) $hyp_{1Avancer} = hyp_{1Reculer} = hyp_{1Descendre} = 0*$

L'INCOHÉRENCE ENLEVÉE, il nous reste qu'à finir l'algorithme pour obtenir les systèmes des sorties modifiés et cohérents. Le système de sorties cohérent est alors

P6_1)

$$\begin{cases} f_{vAvancer} = Phaute \cdot Pprise \cdot (\overline{ARU} \cdot \overline{Pdepose}) \\ g_{vAvancer} = ARU \cdot \overline{Pbasse} \cdot \overline{Pdepose} + \overline{Phaute} + \overline{Pprise} \cdot (ARU + Pdepose) \end{cases}$$

$$\begin{cases} f_{vReculer} = Phaute \cdot Pdepose \cdot \overline{ARU} \cdot \overline{Pprise} \\ g_{vReculer} = \overline{Pdepose} \cdot Pprise + \overline{Phaute} + ARU \cdot \overline{Pbasse} \cdot \overline{Pprise} \\ h_{vReculer} = \overline{ARU} \cdot Phaute \cdot \overline{Pdepose} \cdot \overline{Pprise} \end{cases}$$

$$\begin{cases} f_{vDescendre} = Phaute \cdot \overline{ARU} \cdot \left(\overline{Pbasse} \cdot (Pdepose + \uparrow dcy \cdot Pprise) + \uparrow dcy \cdot Pprise \cdot \overline{Pdepose} \right) \\ g_{vDescendre} = \begin{pmatrix} Pbasse \cdot Pdepose \cdot \overline{Phaute} + 1s/(P\_prise \cdot P\_basse) + \overline{Pdepose} \cdot \overline{Pprise} \\ +ARU \cdot \left(\overline{\uparrow dcy} \cdot \overline{Pdepose} + \overline{Phaute} + \overline{Pbasse} \cdot (\overline{Pdepose} + \overline{Pprise}) \right) \end{pmatrix} \end{cases}$$

$$\begin{cases} f_{vAspirer} = Pbasse \cdot Pprise \\ g_{vAspirer} = Pbasse \cdot Pdepose \end{cases}$$

$$\begin{cases} h_{vAvancer} = h_{vDescendre} = h_{vAspirer} = 0* \end{cases}$$

8.1.2.2 Vérification et corrections des incohérences dues aux états interdits

La commande demande à interdire les états suivants qui doivent être inaccessibles :

$$Avancer \cdot Reculer + Avancer \cdot Descendre + Reculer \cdot Descendre = 0*$$

Il suffit d'interdire chaque état pour que l'ensemble des états soit interdit (voir [2.25]). En effet, il suffit de prouver que $\uparrow (Avancer \cdot Reculer) = 0*$, $\uparrow (Avancer \cdot Descendre) = 0*$ et $\uparrow (Reculer \cdot Descendre) = 0*$. A partir de l'équation développée au chapitre précédent nous déterminons la propriété à vérifier pour chaque terme qui a deux sorties. Les résultats après substitution et simplification sont les suivants :

$$hyp_{fmAR} = 0*$$

$$hyp_{fmAD} = \uparrow dcy \cdot Phaute \cdot Pprise \cdot \overline{Pbasse} \cdot \overline{Pdepose} \cdot \overline{ARU} \cdot (\overline{Avancer} + \overline{Descendre})$$
$$hyp_{fmRD} = Phaute \cdot Pdepose \cdot \overline{Pbasse} \cdot \overline{Pprise} \cdot \overline{ARU} \cdot (\overline{Reculer} + \overline{Descendre})$$

Nous utiliserons donc l'algorithme 2, en considérant Q5-R2, pour corriger les deux dernières propriétés car la première est respectée. Nous n'allons montrer ici que les pas principaux de la démarche sans montrer tous les calculs et résultats.

P1_1) Nous avons déjà calculés le respect des états interdits à partir de la SF introduite par l'utilisateur. Les résultats sont donc hyp_{fmAD} et hyp_{fmRD} (voir ci-dessus). La propriété (**Q1_1**) n'est pas respecté pour $Avancer \cdot Descendre$ et $Reculer \cdot Descendre$ car ces états sont accessibles depuis l'ensemble des états accessibles.

Q4_R2) Il est nécessaire de corriger la SF, les modifications cependant doivent être faites par l'utilisateur sur les parties qui conduisent le système vers un état interdit : hyp_{fmAD} et hyp_{fmRD}. L'utilisateur doit donc modifier la SF car elle est contraignante et par conséquence sans solution. En effet, à partir du hyp_{fmAD} nous mettons en évidence que l'utilisateur demande d'avancer et aussi de reculer dans la SF mais il ne faut pas en plus avancer et reculer en même temps. C'est la même chose avec hyp_{fmRD} car il demande de reculer et aussi de descendre dans la SF mais il ne faut pas en plus reculer et descendre en même temps.

La question à poser à l'utilisateur est donc $\begin{matrix} hyp_{fmAD} \leq \{Avancer, Descendre\} ? \\ hyp_{fmRD} \leq \{Reculer, Descendre\} ? \end{matrix}$

Analyse du système :

> Pour $Avancer$ et $Descendre$, la réponse paraît évident : il faut descendre car hyp_{fmAD} est la condition pour commencer le cycle. Cependant, supposons le cas où le cycle est déjà commencé et que le préhenseur est déjà pris la pièce, il soit au $Phaute \cdot Pprise \cdot \overline{Pdepose}$, le bouton d'urgence ne soit pas activé et que le bouton *dcy* (début du cycle) soit appuyé. Dans ce cas là, le cycle doit continuer et le préhenseur avancer.
>
> Nous remarquons en plus que la commande ne dépend pas de l'état précédent d' $Avancer$ et de $Descendre$ car dans les deux cas de figure (au début du cycle et au moment d'avancer le préhenseur) l'état précédent est $\overline{Avancer} \cdot \overline{Descendre}$. ***Cela veut dire que l'utilisateur ne doit pas choisir s'il faut Avancer ou Descendre avec*** hyp_{fmAD}***, il doit plutôt préciser la commande.***

D'autre part, pour *Descendre* et *Reculer* la condition hyp_{fmRD} montre aussi que la commande est ambiguë. En effet, au $Phaute \cdot Pdepose \cdot \overline{Pbasse} \cdot \overline{Pprise}$ et le bouton d'urgence n'est pas activé, le préhenseur doit descendre pour déposer la pièce quand elle est prise et par contre, une fois que la pièce est déjà déposé le préhenseur doit reculer afin de finir le cycle. Nous remarquons en plus que la commande ne dépend pas de l'état précédent de *Reculer* et de *Descendre* car dans les deux cas de figure (le préhenseur avec et sans la pièce) l'état précédent est $\overline{Reculer} \cdot \overline{Descendre}$. ***Cela veut dire que l'utilisateur ne doit pas choisir s'il faut Descendre ou Reculer avec*** hyp_{fmRD}***, il doit plutôt préciser la commande.***

☞ **REPONSE :**

Notons la différence dans les deux cas analysés et, le fait que la pièce soit ou non prise. Or, depuis les assertions *A2* et *A6* il s'avère que la pièce est prise par la commande *Aspirer*. De ce fait, la commande peut être précisée à partir des assertions suivantes. Notons que la sortie *Aspirer* sert à définir le choix entre une option et l'autre (A9 et A11; A10 et A12; A13 et A15; et A14 et A16) :

$$\begin{cases} \begin{cases} A17) & \left(Phaute \cdot Pprise \cdot \overline{ARU}\right) \cdot \overline{Aspirer} \leq \overline{Avancer} \\ A18) & \left(\uparrow dcy \cdot Phaute \cdot Pprise \cdot \overline{ARU}\right) \cdot \overline{Aspirer} \leq Descendre \end{cases} \\ \begin{cases} A19) & \left(Phaute \cdot Pprise \cdot \overline{ARU}\right) \cdot Aspirer \leq Avancer \\ A20) & \left(Phaute \cdot Pprise \cdot \overline{ARU}\right) \cdot Aspirer \leq \overline{Descendre} \end{cases} \\ \begin{cases} A21) & (Phaute \cdot Pdepose \cdot \overline{ARU}) \cdot Aspirer \leq \overline{Reculer} \\ A22) & (Phaute \cdot Pdepose \cdot \overline{ARU}) \cdot Aspirer \leq Descendre \end{cases} \\ \begin{cases} A23) & (Phaute \cdot Pdepose \cdot \overline{ARU}) \cdot \overline{Aspirer} \leq Reculer \\ A24) & (Phaute \cdot Pdepose \cdot \overline{ARU}) \cdot \overline{Aspirer} \leq \overline{Descendre} \end{cases} \end{cases}$$

La somme de ces assertions nous donne la réponse de l'utilisateur :

$k_{fAvancer}$, $k_{gAvancer}$, $k_{fReculer}$, $k_{gReculer}$, $k_{fReculer}$, $k_{gReculer}$ $k_{fDescendre}$ $k_{gDescendre}$

dont $k_{hAvancer} = k_{hReculer} = k_{fDescendre} = U_{0a} = 0$ *

Remarques :

- *Nous soulignons en plus que nous avons ajouté d'assertions et non pas de priorités à respecter. Ceci afin de mettre en évidence la correction à faire dans l'algorithme 2 car ces nouvelles assertions sont évidemment incohérentes par rapport à la SF de départ.*
- *D'autre part, il faut souligner qu'il n'est pas considéré* $\uparrow dcy$ *comme facteur dans les assertions A17, A19 et A20 car celles-ci seraient plus contraignantes qu'elles doivent l'être.*
- *La considération de ce terme aurait eu en plus comme conséquence, après la correction de l'incohérence, l'introduction implicite de* $\overline{\uparrow dcy} \cdot Phaute \cdot Pprise \cdot \overline{Pdepose} \cdot \overline{ARU} \leq Avancer$. *Cette assertion n'est pas correcte et celle-ci met en évidence que les corrections dans cette partie sont délicates et l'utilisateur doit être attentif aux corrections qu'il fait.*

Q5-R1_1) Il faut faire la vérification de la cohérence du système avec l'introduction des assertions A17 à A24. Nous continuons donc cette vérification en utilisant l'algorithme 1.

Nous commençons la vérification, au premier cycle, par le calcul de la propriété de la cohérence : $hyp_1 = \left(k_f \cdot k_g + k_f \cdot k_h + k_g \cdot k_h\right) + \left(f \cdot (k_g + k_h) + g \cdot (k_f + k_h) + h \cdot (k_f \cdot k_g)\right) = 0*$,

Le résultat nous indique que l'ajout des spécifications données ci-dessus n'est pas cohérent avec les systèmes des sorties et ses propriétés.

Cela veut dire que les systèmes des sorties : *Avancer*, *Reculer* et *Descendre,* sont incohérents car les sorties ne sont pas égaux à 0*. Ce résultat était attendu car ces assertions ont été ajoutées comme corrections. Ainsi, en suivant l'algorithme 1, l'utilisateur répondra lui-même aux incohérences :

Au Q4_R2 nous recevons comme réponse :

$$K_{gAvancer} \quad \Rightarrow \quad Phaute \cdot Pprise \cdot \overline{Pbasse} \cdot \overline{Pdepose} \cdot \overline{ARU} \cdot \overline{Aspirer} \leq \overline{Avancer}$$

$$K_{gReculer} \quad \Rightarrow \quad Phaute \cdot Pdepose \cdot \overline{Pbasse} \cdot \overline{Pprise} \cdot \overline{ARU} \cdot Aspirer \leq \overline{Reculer}$$

$$K_{gDescendre} \quad \Rightarrow \quad Phaute \cdot \overline{Pbasse} \cdot \overline{ARU} \cdot \left(\begin{array}{r} \uparrow dcy \cdot Pprise \cdot \overline{Pdepose} \cdot Aspirer \\ +Pdepose \cdot \overline{Pprise} \cdot \overline{Aspirer} \end{array} \right) \leq \overline{Descendre}$$

$$K_{fAvancer} = K_{fReculer} = K_{fDescendre} = K_{hAvancer} = K_{hReculer} = K_{hDescendre} = U_{0a} = 0*$$

Un nouveau cycle de calcul de vérification de la cohérence commence et à la fin nous constatons le respect de la cohérence, nous obtenons le système cohérent introduit pour résoudre aux états interdits :

P6_2)

$$\begin{cases} f_{vAvancer} = Phaute \cdot Pprise \cdot \overline{ARU} \cdot \overline{Pdepose} \cdot (Aspirer + Pbasse) \\ g_{vAvancer} = \left(\begin{array}{l} \overline{Phaute} + (ARU + Pprise \cdot \overline{Aspirer}) \cdot \overline{Pbasse} \cdot \overline{Pdepose} \\ \quad + \overline{Pprise} \cdot (ARU + Pdepose) \end{array} \right) \\ h_{vAvancer} = 0* \end{cases}$$

$$\begin{cases} f_{vReculer} = Phaute \cdot Pdepose \cdot \overline{Pprise} \cdot \overline{ARU} \cdot (\overline{Aspirer} + Pbasse) \\ g_{vReculer} = \left(\begin{array}{l} \overline{Pdepose} \cdot Pprise + \overline{Phaute} \\ \quad + (ARU + Aspirer \cdot Pdepose) \cdot \overline{Pbasse} \cdot \overline{Pprise} \end{array} \right) \\ h_{vReculer} = \overline{ARU} \cdot Phaute \cdot \overline{Pdepose} \cdot \overline{Pprise} \end{cases}$$

$$\begin{cases} f_{vDescendre} = Phaute \cdot \overline{ARU} \cdot \left(\begin{array}{l} Pdepose \cdot \overline{Pbasse} \cdot (Aspirer + Pprise) \\ \quad + \uparrow dcy \cdot Pprise \cdot \overline{Pdepose} \cdot (\overline{Aspirer} + Pbasse) \end{array} \right) \\ g_{vDescendre} = \left(\begin{array}{l} 1s/(Pprise \cdot Pbasse) + \overline{Phaute} \cdot (ARU + Pbasse \cdot Pdepose) \\ \quad + \overline{Pbasse} \cdot \left(\begin{array}{l} \overline{Pdepose} \cdot (ARU + Aspirer \cdot Phaute) \\ + \overline{Pprise} \cdot (ARU + \overline{Aspirer} \cdot Phaute) \end{array} \right) \\ \quad + \overline{Pdepose} \cdot \overline{Pprise} + \uparrow \overline{dcy} \cdot \overline{Pdepose} \cdot ARU \end{array} \right) \\ h_{vDescendre} = 0* \end{cases}$$

$$\begin{cases} f_{vAspirer} = Pbasse \cdot Pprise \\ g_{vAspirer} = Pbasse \cdot Pdepose \\ h_{vAspirer} = 0* \end{cases}$$

Après avoir vérifié la cohérence des assertions introduites nous continuons l'algorithme 2 au P5 afin de commencer le deuxième cycle en considérant le dernier système introduit (*P6_2*).

P1_2) Calcul du respect des états interdits avec la SF introduite par l'utilisateur (P6_2):

$$hyp_{fmAD} = \uparrow dcy \cdot Pbasse \cdot Phaute \cdot Pprise \cdot \overline{Pdepose} \cdot \overline{ARU}$$
$$hyp_{fmRD} = 0*$$

$P_{u0\_1\_2}$) L'état $Avancer \cdot Descendre$ n'est pas accessible qu'en cas de panne des capteurs. En effet, en considérant la SF-T3, notamment A10 ($Pbasse \cdot Phaute = 0*$), nous avons

$$hyp_{eiADav} = hyp_{eiAD} \cdot \overline{Uo} = 0*$$

Nous soulignons que cet état peut être accessible si la panne des capteurs arrive. C'est-à-dire, si le signal $\uparrow dcy \cdot \overline{Pdepose} \cdot \overline{ARU} \cdot (Pbasse \cdot Phaute \cdot Pprise)$ devient vrai à un instant t quelconque alors le préhenseur avancera et descendra au même temps.

Nous finissons donc la vérification des états interdits car nous venons de constater que ces états ne sont pas accessibles en considérant que la panne des capteurs n'arrive jamais.

8.1.2.3 Vérification et correction de la SF afin que le système soit unité stable

Nous avons déterminé qu'une spécification partielle respecte l'unité stabilité si $hyp_{us} = 0*$ où :

$$hyp_{us} = \left(\prod_{i \in \{1,p\}} (f_{xi} + h_{xi} \cdot X_i) \right) \cdot \overline{fouh_{etat}} \text{ tel que } fouh_{etat} = \left(\prod_{i \in \{1,p\}} (f_{xi} + h_{xi} \cdot X_i) \right) \Big|_{\left(\prod_{i \in \{1,p\}} X_i \right) = 1^*}$$

Après calcul, nous obtenons pour huit états non interdits (tel que $\overline{Ava} \cdot \overline{Rec} \cdot \overline{Des} \cdot \overline{Asp}$) :

$$hyp_{us}(\overline{Ava} \cdot \overline{Rec} \cdot \overline{Des} \cdot \overline{Asp}) = hyp_{us}(\overline{Ava} \cdot \overline{Rec} \cdot Des \cdot \overline{Asp}) = hyp_{us}(\overline{Ava} \cdot Rec \cdot \overline{Des} \cdot \overline{Asp}) = hyp_{us}(Ava \cdot \overline{Rec} \cdot \overline{Des} \cdot \overline{Asp}) = 0^*$$
$$hyp_{us}(\overline{Ava} \cdot \overline{Rec} \cdot \overline{Des} \cdot Asp) = hyp_{us}(\overline{Ava} \cdot \overline{Rec} \cdot Des \cdot Asp) = hyp_{us}(\overline{Ava} \cdot Rec \cdot \overline{Des} \cdot Asp) = hyp_{us}(Ava \cdot \overline{Rec} \cdot \overline{Des} \cdot Asp) = 0^*$$

Cela veut dire que pour tous les états non interdits, d'après cette spécification partielle, l'unité-stabilité est respectée. Nous remarquons cependant que ce résultat est nécessaire mais il n'est pas encore suffisant pour garantir l'unité-stabilité car les systèmes ne sont pas complets.

8.1.3 Résolution de l'incomplétude

Pour cet exemple, nous avons choisi de grouper les pas du calcul des conditions d'existence d'une solution complète et la résolution de l'incomplétude car avec le choix prédéterminé de rester dans l'état précédent, il est possible de substituer ce choix et vérifier si les propriétés demandées sont respectées. Nous allons donc suivre la démarche décrite au chapitre 6.

1) Nous calculerons donc les conditions d'existence en suivant le choix de rester dans l'état précédent (tel que l'exemple 2) :

$k_{favancer} = k_{gavancer} = k_{freculer} = k_{greculer} = k_{fdescendre} = k_{gdescendre} = k_{faspirer} = k_{gaspirer} = 0^*$,

$k_{havancer} = \overline{f_{avancer}} \cdot \overline{g_{avancer}} \cdot \overline{h_{avancer}}$, $k_{hreculer} = \overline{f_{reculer}} \cdot \overline{g_{reculer}} \cdot \overline{h_{reculer}}$,

$k_{hdescendre} = \overline{f_{descendre}} \cdot \overline{g_{descendre}} \cdot \overline{h_{descendre}}$ et $k_{haspirer} = \overline{f_{aspirer}} \cdot \overline{g_{aspirer}} \cdot \overline{h_{aspirer}}$,

2) Comme les états accessibles ne sont pas bloquants, alors nous vérifierons le respect des états interdits. En effet, comme toute l'incomplétude reste dans l'état précédent alors la cohérence des sorties vérifiée au 8.1.2 est toujours respectée.

Ainsi, nous allons calculer le respect des états interdits, tel que nous l'avons fait dans la section 8.1.2.2 mais pour le système complété avec le choix de rester dans l'état précédent.

Il s'avère que $hyp_{fmtotAR} = hyp_{fmtotAD} = hyp_{fmtotRD} = 0^*$. Par conséquence le choix de rester dans l'état précédent respect les états interdits et aucune assertion additionnelle est nécessaire.

3) Il n'est pas nécessaire de revérifier l'unité stabilité car il n'y a pas eu aucun ajout d'assertions au point 2.

Cela veut dire que le système respect toutes les propriétés en considérant que l'incomplétude reste dans l'état précédent.

En considérant la SF-T3, l'expression du système solution en simplifiant en utilisant [2.25] et les équivalences [2.34] à [2.41]; les fonctions des sorties *Avancer*, *Reculer* et *Descendre* peuvent se simplifier comme suit :

$$\begin{cases} f_{Avancer} = Phaute \cdot Pprise \cdot \overline{ARU} \cdot Aspirer \\ g_{Avancer} = ARU + Pdepose + \overline{Phaute} + Pprise \cdot \overline{Aspirer} \end{cases}$$

$$\begin{cases} f_{Reculer} = Phaute \cdot Pdepose \cdot \overline{ARU} \cdot \overline{Aspirer} \\ g_{Reculer} = ARU + Pprise + \overline{Phaute} + Pdepose \cdot Aspirer \end{cases}$$

$$\begin{cases} f_{Descendre} = Phaute \cdot \overline{ARU} \cdot \left(\uparrow dcy \cdot Pprise \cdot \overline{Aspirer} + Pdepose \cdot Aspirer \right) \\ g_{Descendre} = \left(\begin{array}{c} ARU + 1s/(Pprise \cdot Pbasse) + Pbasse \cdot Pdepose + \overline{Pdepose} \cdot \overline{Pprise} \\ + Phaute \cdot (Pdepose \cdot \overline{Aspirer} + \overline{Pdepose} \cdot Aspirer) \end{array} \right) \end{cases}$$

$$\begin{cases} SF-T5) \quad \uparrow (Avancer \cdot Reculer) + \uparrow (Avancer \cdot Descendre) + \uparrow (Reculer \cdot Descendre) = 0 * \\ SF-T6) \quad Pbasse \cdot Phaute + Pprise \cdot Pdepose = 0 * \end{cases}$$

Les fonctions des sorties à la sortie du calculateur sont les suivants :

$$Avancer = SR(f_{vAvancer}, g_{vAvancer}) \qquad Descendre = SR(f_{vDescendre}, g_{vDescendre})$$
$$Reculer = SR(f_{vReculer}, g_{vReculer}) \qquad Aspirer = SR(f_{vAspirer}, g_{vAspirer})$$

Nous précisons pour l'implantation en parallèle de section suivante que

$X_i = f_i + \overline{g_i} \cdot x_i = f_i + h_i \cdot x_i = f_i + (f_i + h_i) \cdot x_i$ où $h = \overline{f} \cdot \overline{g}$.

8.1.4 Implantation

L'implantation sur un calculateur demande en plus la semi-insensibilité à l'ordre. La figure 44 montre le graphe de dépendance. Nous remarquons que le calcul d'*Aspirer* peut affecter le calcul d'*Avancer*, *Reculer* et *Descendre* mais *Avancer*, *Reculer* et *Descendre* n'ont aucune influence sur *Aspirer*. En conséquence le système est insensible à l'ordre en calculant d'abord *Avancer*, *Reculer* et *Descendre* et après *Aspirer*. Les sorties *Avancer*, *Reculer* et *Descendre* peuvent se calculer en n'importe quel ordre car elles sont indépendantes entre elles. D'autre part, la *variable interne* $1s/(Pprise \cdot Pbasse)$ ne dépend que des entrées et elle affecte *Descendre*, par conséquence elle doit être calculée avant *Descendre* afin de ne pas provoquer la non unité stabilité. En effet, on arrive à l'état désiré de *Descendre* en un seul pas de calcul.

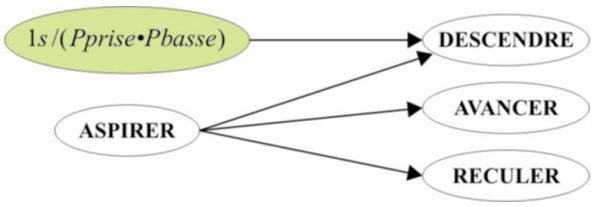


Figure 44 *Graphe de dépendances*

La figure 45 montre le diagramme d'implantation sur : a) un calculateur exécuté en séquentiel qui respecte la semi-insensibilité à l'ordre et b) sur un calculateur exécuté en parallèle

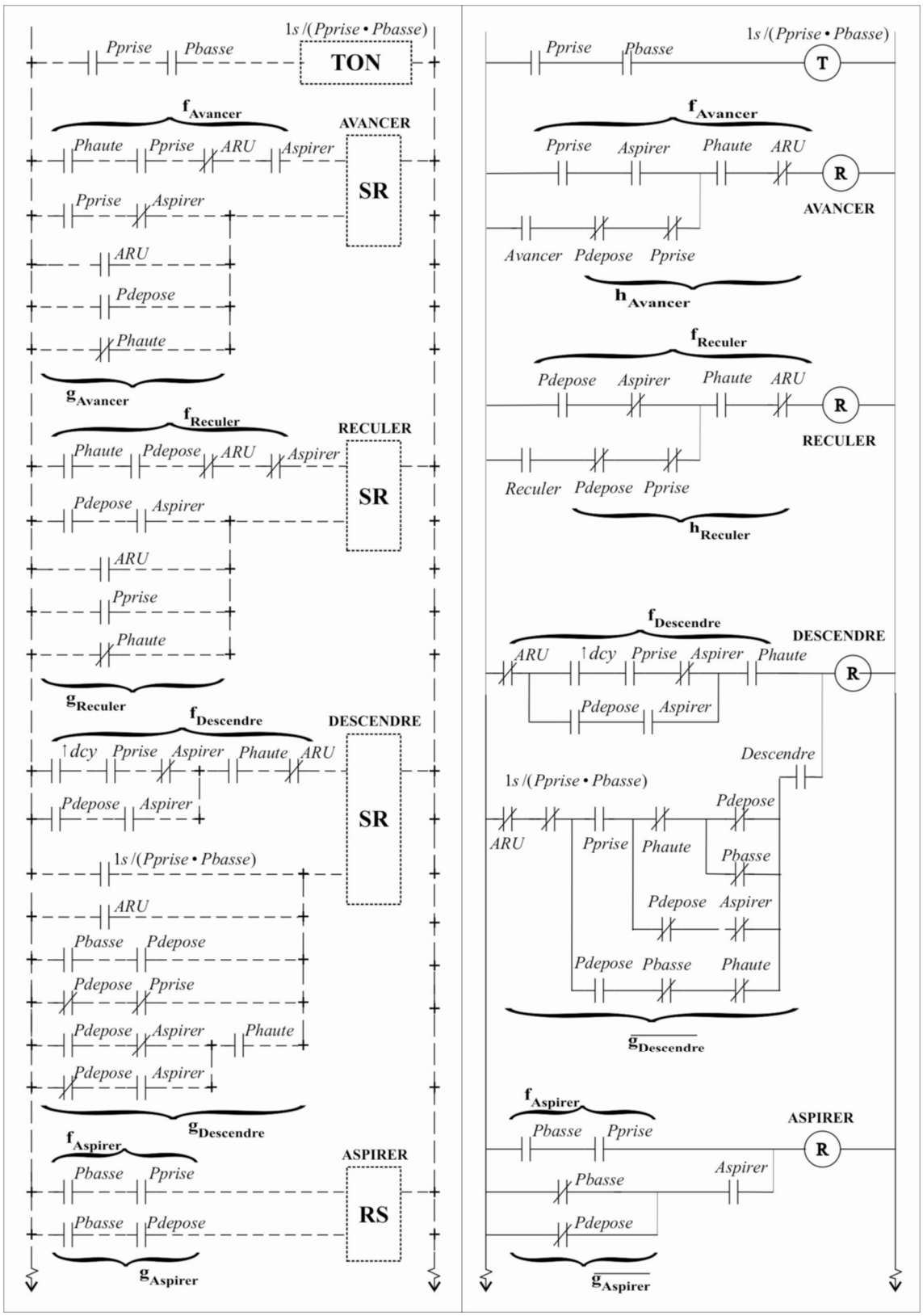


Figure 45 *Diagramme d'implantation des fonctions de sortie de l'exemple 3 a) Diagramme Ladder et b) diagramme câblé*

8.2 **Analyse du troisième exemple**

Cet exemple nous a permis de montrer plusieurs aspects de notre proposition :

D'une part, nous avons utilisé des théorèmes développés dans les chapitres précédents, notamment le TON. Ces théorèmes nous permettent de vérifier formellement des propriétés (toujours vraies ou toujours fausses) où l'utilisation d'opérations temporelles est nécessaire.

D'autre part, nous a permis l'utilisation des algorithmes sur un cas plus complexe où plusieurs sorties sont utilisées.

La spécification donnée par l'utilisateur n'était pas au point cependant l'exemple a permis de mettre en évidence que la méthode nous permet de surmonter cette problématique en faisant appel à l'utilisateur.

Nous précisons que cet exemple a été choisi afin de montrer la résolution pour un système à plusieurs sorties jusqu'à l'implantation sur un calculateur qui exécute les fonctions séquentiellement. La validation de la semi-insensibilité à l'ordre est montrée à travers un graphe de dépendances.

8.3 **Conclusion**

Cet exemple nous a permis de montrer l'utilisation des algorithmes sur un cas plus complexe où il y a plusieurs sorties. La spécification donnée par l'utilisateur n'était pas au point cependant la méthode nous a permis d'identifier les causes du non-respect dans la SF. La démarche proposée indique les pas à suivre pendant la synthèse pour arriver à calculer les fonctions de contrôle. Néanmoins, c'est l'expérience et le savoir faire de l'utilisateur qui permettra d'analyser les incohérences et surmonter les incohérences en ajoutant ou en enlevant des assertions.

Si la SF est correctement spécifiée dès le départ, la méthode proposée permettra de générer automatiquement la fonction de contrôle en considérant les propriétés que nous avons défini. Les algorithmes de résolution ont été conçus cependant en considérant que dans la pratique industrielle les erreurs et les oublis dans la phase de conception peuvent arriver souvent.

Conclusions et Perspectives

Le travail présenté tout au long de ce mémoire a consisté au développement d'une méthode alternative à la Théorie de la Commande Supervisée. Le souci majeur de notre démarche a été de garantir a priori la sûreté de fonctionnement de la commande et le but principal, l'application industrielle de la méthode.

Le travail de recherche ne pouvait commencer que par la définition du cadre formel. L'algèbre $\mathbb{I}$ étant défini et utilisée pour la vérification formelle [Rou 03b] nous a offert des bases formelles très intéressantes à notre recherche. Pour pouvoir utiliser cette algèbre dans la synthèse nous avons dû, dans un premier temps, participer à la définition et à la démonstration des théorèmes présentés dans ce mémoire.

A ce sujet, nous avons contribué à l'ajout de certains théorèmes de l'algèbre de base et des opérateurs TON, TOF, FE et RE ainsi qu'aux démonstrations d'un total de 53 théorèmes. Nous avons dû redéfinir le SR car il présentait des imprécisions dans la définition et par conséquence tous les théorèmes ont été donc redémontrés (28 théorèmes). La partie la plus importante, de notre contribution est portée sur la relation d'ordre partiel et son utilisation pour la résolution d'équations (62 théorèmes).

Le premier apport de nos travaux découle de cette participation. En effet, nous avons observé que la relation d'ordre pouvait représenter la spécification de fonctionnement de la commande. Ce principe a été utilisé pour la vérification formelle. Nous avons cependant proposé l'utilisation de cette relation d'ordre pour la définition d'un jeu d'équations représentant la SF. *Nous nous sommes proposés comme objectif la résolution de ce système d'équations, représentant la SF, afin de calculer la fonction de contrôle.*

Ceci nous a permis de *proposer une méthode de synthèse basée sur des signaux algébriques* mais aussi de couvrir plusieurs phases de cycle de vie de contrôleurs logiques (spécification,

conception et implantation) *en utilisant un seul cadre mathématique*, l'algèbre $\mathbb{I}$, de telle sorte à conserver la cohérence de formalisme jusqu'à l'implantation sur un contrôleur logique.

Nous pouvons citer plusieurs contributions qui ressortent de ce mémoire de thèse :

— Nous avons présenté une technique de résolution de système d'équations.

— Ce système d'équations, représentant une spécification de fonctionnement, nous a permis de présenter une méthode de synthèse en utilisant cette technique de résolution.

— Nous avons défini formellement sur $\mathbb{I}$ des propriétés à respecter : les deux propriétés inhérentes à la solution d'un système d'équations : la cohérence et la complétude et en plus, d'autres propriétés inhérentes aux systèmes de commande : le non blocage, l'unité stabilité et la semi-insensibilité à l'ordre. Nous avons considéré aussi les états interdits.

— Nous avons proposés des algorithmes de correction.

Nous envisageons comme perspectives, notamment :

- Continuer le développement du programme crée en Mathématica pour l'analyse, le calcul et la synthèse des expressions algébriques. Le but serait de systématiser la méthode.
- La recherche des expressions formelles, en algèbre $\mathbb{I}$, décrivant des comportements séquentiels. Le premier objectif serait de représenter des contraintes d'évolution des signaux d'entrée. Par exemple, ne pas attendre la détection de fin de sortie d'un vérin si un capteur indique qu'il est rentré (début de course).
- La recherche des expressions de vérification des propriétés. Surtout, celles qui décrivent des contraintes sur le comportement global du système tel que la vivacité.

Ces recherches peuvent servir, entre autre, pour étudier les possibilités de faire une passage a l'échelle et aussi d'avoir des moyens de comparaison avec des exemples traités surtout par la théorie de Ramadge and Wonham qui est la technique de synthèse de la commande la plus utilisée dans la communauté scientifique. Ces comparaissons sont indispensables pour démontrer que la méthode que nous proposons peut surmonter la limitation de l'explosion combinatoire.

Bibliographie

[Bal 92]	BALEMI S., *Control of Discrete Event Systems : Theory and Application*, Thesis for the degree of Doctor of Technical Sciences, Swiss Federal Institute of Technology, Zurich, Mai 1992.
[Bal 93]	BALEMI S., HOFFMANN G. J., GYUGYI P., WONG-TOI H., FRANKLIN G. F., *Supervisory control of a rapid thermal multiprocessor*, IEEE Transaction on Automatic Control, Vol. 38, No. 7, pp. 1040-1059, July 1993.
[Ber 91]	BERLIOUX P., LEVY M., "Théorie des langages", *Notes de cours*, ENSIMAG, Mars 1991.
[Bou 92]	BOUTEILLE N., BRARD P., COLOMBARI G., COTAINA N., RICHET D., *Le GRAFCET,* Toulouse, Editions Cépaduès, 1992, ISBN 2-85428-307-4.
[Bra 89]	BRANDIN B.A, *The Supervisory Control of Discrete Event Systems with Forcible Events*, Thesis for the degree of Master of Applied Science, Departement of Electrical Engineering, Université de Toronto, Octobre 1989.
[Cas 98]	CASPI P., SALEM R., ALLA H., DAVID R., YEDDES M., *Communicating periodic synchronous processes*, Technical report, LAG 98-160, CRISYS Project EP 25.514, December 1998.
[Cas 99]	CASSANDRAS C. G., LAFORTUNE S., *Introduction to Discrete Event Systems*, Boston Kluwer Academic Publishers, 1999.
[Cha 96]	CHARBONNIER F., *Commande supervisée des Systèmes à Evénements Discrets*, Thèse de doctorat de l'Institut National Polytechnique de Grenoble, Spécialité : Automatique - Productique, 10 Janvier 1996.
[Did 06]	DIDEBAN A., ALLA H., *Solving the Problem of Forbidden States by Feedback Control Logical Synthesis*, IEEE Industrial Electronics, IECON 2006 – 32nd Annual Conference, pp. 348-353, November 2006, Paris, FRANCE.
[Fab 98]	FABIAN M., HELLGREN A., *PLC-based Implementation of supervisory control for discrete event systems*. Proceeding of the 37th IEEE Conference on Decision and Control, Vol. 3, pp. 3305-3310, Tampa, Florida, 1998.
[Fau 01]	FAURE J.M., LESAGE J.J., *Methods for safe control systems design and implementation*, 10th IFAC Symposium on Information Control Problems in Manufacturing, INCOM 2001, Vienna, AUSTRIA, 20-22 September 2001.
[Gaf 03]	GAFFE D., *Modélisation et vérification d'un système mécatronique par SyncCharts*, Actes du Congrès Modélisation des Systèmes Réactifs, MSR 2003, pp. 95-107, Metz, FRANCE, 6-8 Octobre 2003.

[Gha 02]	GHAFFARI A., *Les réseaux de Petri pour la synthèse de contrôle des systèmes à événements discrets,* Thèse de doctorat, Université de Metz, Décembre 2002.
[Goh 98]	GOHARI P., WONHAM W.M., *A linguistic framework for controlled hierarchical DES*, Proceedings of the 4th Workshop On Discrete Event Systems, WODES 98, pp. 207-212, Cagliari, ITALY, 26-28 August 1998.
[Gou 02]	GOUYON D., GOUIN A, PETIN J.F., *Application de techniques de synthèse en ingénierie d'automatisation*, Conférence Internationale Francophone d'Automatique, CIFA 2002, pp. 62-67, Nantes, FRANCE, 08-10 juillet 2002.
[Gou 03]	GOUYON D., PETIN J.F., GOUIN A., *Modèles du procédé et de ses spécifications pour la synthèse de la commande*, Actes du Congrès Modélisation des Systèmes Réactifs, MSR 2003, pp. 45-60, Metz, FRANCE, 6-8 Octobre 2003.
[Gou 04]	GOUYON D., *Contrôle par le produit des systèmes d'exécution de la production : apport des techniques de synthèse*, Thèse de doctorat de l'Université Henri Poincaré, Nancy-I, Spécialité : Automatique, Traitement du signal, Génie informatique, 6 Décembre 2004.
[Gri 00]	GRIMALDI R. P., *Discrete and Combinatorial Mathematics : an applied introduction*, Reading Addison-Wesley, 4th edition, 2000, ISBN : 0-201-19912-2.
[Han 95]	HANISH H.M., RAUSH M., *Synthesis of Supervisory controllers based on novel representation of condition/event systems*, IEEE International Conference on Systems, Man and Cybernetics, pp. 3069-3074, Vancouver, British Columbia, CANADA, 22-25 October 1995.
[Han 97]	HANISCH H.-M., THIEME J., LUDER A., *Towards a synthesis method for distributed safety controllers based on net condition/event systems*, Journal of Intelligent Manufacturing, Vol. 8, No. 5, pp. 357-368, September 1997.
[Hel 02]	HELLGREN A., LENNARTSON B., FABIAN M., *Modelling and PLC-based implementation of modular supervisory control,* Proceedings of the 6th International Workshop On Discrete Event Systems, WODES 2002, pp. 371-376, Zaragoza, SPAIN, 2-4 October 2002.
[Hop 79]	HOPCROFT J.E. AND ULLMAN J. D., *Introduction to automata theory, languages and computation*, Reading, MA: Addison-Wesley, 1979.
[IEC 61131-3]	International Electrotechnical Commission: Programmable controllers, Part3 – Programming languages, IEC 61131-3, Genève, 1993.
[Kat 04]	KATTAN B., *Synthèse structurelle d'un contrôleur basée sur le Grafcet*, Thèse de doctorat de l'Université Joseph Fourier (Grenoble), Spécialité : Automatique-Productique, 3 Septembre 2004.
[Kap 94]	KAPELLOS K. *Environnement de programmation des applications robotiques réactives*, Thèse de doctorat de l'Ecole de Mines de Paris : Informatique temps réel, robotique, automatique, 1 Novembre 1994.

[Kum 91]	KUMAR R., GARG V. K., *Modeling and Control of Logical Discret Event Systems*, Kluwer Academic Publishers, Norwell, USA, 1995.
[Kum 96]	KUMAR R., SHAYMAN M. A., *Nonblocking supervisory control of nondeterministic systems via prioritized synchronization*, IEEE Transactions on Automatic Control, Vol. 41, No. 8, pp. 1160-1175, August 1996.
[Laf 01]	LAFORTUNE S., *Control of Partially-Observed Discret-Event System : The State of the Art and Some New Results*, SCODES 2001 – Paris, July 2001.
[Lau 96]	LAUZON S. C., MA A. K. L., MILLS J. K., BENHABIB B., *Application of discrete-event system theory to flexible manufacturing*, IEEE Control Systems Magazine, Vol. 16, No.1, pp. 41-48, February 1996.
[Li 91]	LI Y., *Control of vector discrete event systems*, Ph.D. thesis: Department of Electrical and Computer Engineering, University of Toronto, July 1991.
[Lin 88]	LIN F., WONHAM W. M., *Decentralized supervisory control of discrete-event systems, Information Sciences*, Vol. 44, No. 3, pp. 199-224, April 1988.
[Liu 02]	LIU J., DARABI H., *Ladder Logic Implementation of Ramadge-Wonham Supervisory Controller,* Proceedings of the 6th International Workshop On Discrete Event Systems, WODES 2002, pp. 383-389, Zaragoza, SPAIN, 2-4 October 2002.
[Mar 97]	MARCHAND H., *Méthodes de synthèse d'automatismes décrits par des systèmes à événements discrets finis*, Thèse de doctorat de l'Université de Rennes 1, Spécialité : Informatique, 3 octobre 1997.
[Mar 98]	MARIKAR M.T., ROTSTEIN G.E., MACCHIETTO S., *An integrated environment for the design of procedural controllers*, Preprints Volume II, 9th Symposium on Information Control Manufacturing, Advances in Industrial Engineering, INCOM 1998, pp. 93-99, Nancy, FRANCE, 24-26 June 1998.
[Med 00]	**MEDINA A.**, *Application de la simulation ternaire dans les systèmes synchrones*, Mémoire de recherche du DEA au Laboratoire d'Automatique de Grenoble (LAG) à l'INP de Grenoble, Automatique-Productique, option productique, 3 juillet 2000.
[Med 03]	**MEDINA A.**, ROUSSEL J.M., *Modélisation d'un système logique séquentiel élémentaire à partir d'une spécification algébrique*, Actes des Journées Doctorales d'Automatique, JDA 2003, Valenciennes, FRANCE, pp. 245-250, 25-27 juin 2003.
[Ndj 99]	NDJAB C., ZAYTOON J., *Synthèse en ligne de la commande à partir du Grafcet*, Actes du Congrès Modélisation des Systèmes Réactifs, MSR 1999, pp. 341-350, Cachan, FRANCE, 24-25 mars 1999.
[Ost 89]	OSTROFF J. S., *Synthesis of controllers for real-time discrete event systems*, Proceedings of the 28th IEEE Conference on Decision and Control, Tampa, Florida, Vol. 1, No. 13-15, pp. 138-144, December 1989.

[Phi 03]	PHILIPPOT A., TAJER A., GELLOT F., CARRE-MENETRIER V., *Synthèse de la commande spécifiée en Grafcet : application à un préhenseur pneumatique*, Actes du Congrès Modélisation des Systèmes Réactifs, MSR 2003, pp. 61-75, Metz, FRANCE, 6-8 Octobre 2003.
[Pie 02]	PIETRAC L., CHAFIK S., REGIMBAL L., *Application de la théorie de la supervision : un exemple de conception de programmes d'API.* Conférence Internationale Francophone d'Automatique, CIFA 2002, pp. 728-733, Nantes, FRANCE, 08-10 juillet 2002.
[Pet 81]	PETERSON J. L., *Petri net theory and the modeling of systems,* Prentice Hall, Englewood Cliffs, N. J., 1981.
[Pin 99]	PINZON L. E., HANISCH H.-M., JAFARI M.A., BOUCHER T., *A Comparative Study of Synthesis Methods for Discrete Event Controllers*, Formal Methods in System Design, 15, 123-167, Kluwer Academic Publishers, Netherlands, 1999.
[Que 02]	QUEIROZ M. H., CURY J.E.R., *Synthesis and Implementation of Local Modular Supervisory Control for a Manufacturing Cell*, Proceedings of the 6th International Workshop On Discrete Event Systems, WODES 2002, pp. 377-382, Zaragoza, SPAIN, 2-4 October 2002.
[Ram 82]	RAMADGE P. J., WONHAM W. M., *Supervisory of discrete Event Processes*, 21st IEEE Conference on Decision and Control, Orlando, Floride, Vol. 21, pp. 1228-1229, December 1982.
[Ram 86]	RAMADGE P.J., *Observability of discrete event systems*, 25th IEEE Conference on Decision and Control, Athens, Vol. 25, pp. 1108-1112, December 1986.
[Ram 87a]	RAMADGE P. J., WONHAM W. M., *Supervisory Control of a Class of Discrete Event Process*, SIAM J. Control and Optimisation, Vol. 25, No. 1, pp. 206-230, January 1987.
[Ram 87b]	RAMADGE P. J., WONHAM W. M., *Modular feedback logic for Discrete Event Systems*, SIAM J. Control and Optimisation, Vol. 25, No. 5, pp. 1202-1218, September 1987.
[Ram 89]	RAMADGE P. J. G., WONHAM W. M., *The Control of Discrete Event Systems*, Proceeding of the IEEE, Vol. 77, No. 1, pp. 81-98, January 1989.
[Rau 95]	RAUSCH M., HANISCH H. M., *Net Condition/event systems with multiple conditions outputs*, Proceedings of Symposium on Emerging Technologies and Factory Automation, ETFA 95, Paris, FRANCE, 10-13 October 1995.
[Rou 93]	ROUSSEL J.M., LESAGE J.J., *Une Algèbre de Boole pour l'approche événementielle des systèmes logiques*, APII-AFCET/CNRS, Editions Hermès, Vol. 27, n°5, 1993, pp. 541-560.
[Rou 03a]	ROUSSEL J.M., **MEDINA A.**, FAURE J.M., *Synthèse de la commande d'un système logique à partir de l'expression algébrique de ses spécifications*, Actes du Congrès Modélisation des Systèmes Réactifs, MSR 2003, pp. 77-93, Metz, FRANCE, 6-8 Octobre 2003.

[Rou 03b]	ROUSSEL J.M., FAURE J.M., LESSAGE J.-J., **MEDINA A.**, *Algebraic approach for dependable logic control systems design*, International Journal of Production Research, IJPR 2004, 42(14), pp. 2859-2876, July 2004.
[Rud 92]	RUDIE K., WONHAM W. M.. "Think globally, act locally : Decentralized supervisory control." IEEE Transaction Automatic Control, Vol. 37, No. 11, pp. 1692-1708, November 1992.
[Sha 95]	SHAYMAN M. A., KUMAR R., *Supervisory control of nondeterministic systems with driven events via prioritized synchronization and trajectory models*, SIAM Journal on Control and Optimization, Vol. 33, No. 2, pp. 469-497, March 1995.
[Taj 03]	TAJER A., PHILIPPOT A., GELLOT F., CARRE-MENETRIER V., *Contribution à l'amélioration de la praticabilité des approches formelles de synthèse de commande*, Actes des Journées Doctorales d'Automatique, JDA 2003, Valenciennes, FRANCE, pp. 239-244, 25-27 juin 2003.
[Taj 05]	TAJER A., *Contribution aux approches formelles de synthèse de commande spécifiée par Grafcet*, Thèse de doctorat de l'Université de Reims Champagne Ardenne, Spécialité : Génie informatique, Automatique et Traitement du signal, 3 Novembre 2005.
[Wol 91]	Wolper P., *Introduction à la calculabilité*, Collection IIA Informatique intelligence artificielle, Paris Inter-Editions, 1991.
[Won 87]	WONHAM W. M., RAMADGE P. J., *On the Supremal Controllable Sublanguage Of a Given Language*, SIAM J. Control and Optimisation, Vol. 25, No. 3, pp. 637-659, May 1987.
[Won 88]	WONHAM W. M., RAMADGE P. J., *Modular Supervisory Control of Discrete-Event Systems*, Mathematics of Control, Signals, and Systems, Vol. 1, No. 1, pp. 13-30, January 1988.
[Won 94]	WONHAM W. M., "Notes on Control of Discrete Event Systems", *Technical Report*, ECE 1636F/1637S, 1994.
[Won 98]	WONG K. C., WONHAM W. M., *Modular control and coordination of discrete-event systems*, Discrete Event Dynamic Systems: Theory and Applications, 8(3), pp. 247-297, October 1998.
[Won 99]	WONHAM W. M., Notes on control of discrete-event systems, ECE 1636F/1637S 1998-99, mise à jour du 01/12/1999, (http://odin.control.toronto.edu/DES).
[Yed 00a]	YEDDES M, ALLA H., DAVID R., *Order-insensitivity in Synchronous Distributed Systems*, Technichal Report, LAG 99-064.
[Yed 00b]	YEDDES M, ALLA H., *Checking Order-insensitivity using ternary simulation in synchronous programs*, IEEE, ISPASS2000, Austin (USA), pp. 52-58, 2000.
[Zho 90]	ZHONG H., WONHAM W. M., *On the consistency of hierarchical supervision in discrete-event systems*, IEEE Transactions on Automatic Control, Vol. 35, No. 10, October 1990.

Annexes

A. La norme IEC-61131-3

A.1 Opérations permettant de décrire des événements

La norme [IEC 61131-3], norme relative aux langages de programmation des API, propose deux blocs fonctionnels, dit de détection de fronts, définis à l'aide d'un bloc de code donné en Structured Text (ST). Cette définition est reprise dans le tableau 1 :

<table>
<tr><th colspan="2">Bloc fonctionnel détecteur de front montant (R_TRIG)</th></tr>
<tr><td>Forme graphique

+---------+
| R_TRIG |
BOOL--|CLK Q|--BOOL
+---------+</td><td>Corps du bloc fonctionnel

FUNCTION_BLOCK R_TRIG
VAR_INPUT CLK: BOOL; END_VAR
VAR_OUTPUT Q: BOOL; END_VAR
VAR M: BOOL; END_VAR
Q := CLK AND NOT M;
M := CLK;
END_FUNCTION_BLOCK</td></tr>
<tr><th colspan="2">Bloc fonctionnel détecteur de front descendant (F_TRIG)</th></tr>
<tr><td>Forme graphique

+---------+
| F_TRIG |
BOOL--|CLK Q|--BOOL
+---------+</td><td>Corps du bloc fonctionnel

FUNCTION_BLOCK R_TRIG
VAR_INPUT CLK: BOOL; END_VAR
VAR_OUTPUT Q: BOOL; END_VAR
VAR M: BOOL; END_VAR
Q := NOT CLK AND NOT M;
M := NOT CLK;
END_FUNCTION_BLOCK</td></tr>
</table>

Tableau 1 Blocs fonctionnels détecteurs de fronts (d'après la norme [IEC_61131-3])

Dans cette définition informelle, la valeur Q de la sortie à un instant donné est construite implicitement à partir de la valeur de l'entrée CLK au même instant et la valeur d'une variable M. Dans le cas du front montant, la valeur de M est celle de l'entrée CLK à l'instant précédent. Dans le cas du front descendant, la valeur de M est la valeur opposée de l'entrée CLK à l'instant précédent. D'un point de vue théorique, nous sommes en présence d'une définition sous forme d'équations récurrentes. Cette définition à l'aide d'une récurrence n'est pas nécessaire lorsqu'il est possible de faire référence aux valeurs antérieures de l'entrée CLK, approche qui a été choisie dans [Rou 93].

A.2 Opérations permettant de décrire un comportement séquentiel

La norme [IEC 61131-3], norme relative aux langages de programmation des API, propose deux blocs fonctionnels qui décrivent des comportements séquentiels. La définition proposée dans cette norme est la suivante (tableau 2) :

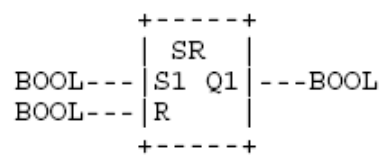
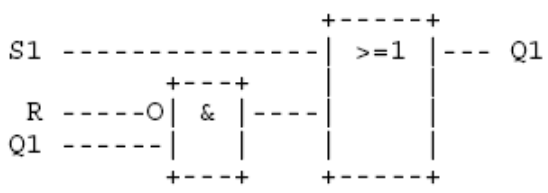
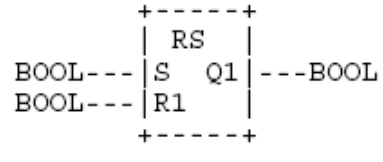
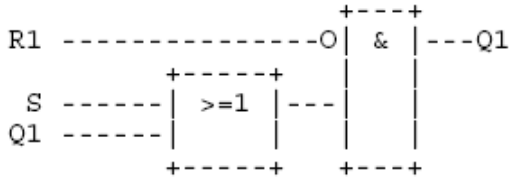
<table>
<tr><td colspan="2">Bloc fonctionnel bistable (Set dominant)</td></tr>
<tr><td>Forme graphique
 
</td><td>Corps du bloc fonctionnel
 
</td></tr>
<tr><td colspan="2">Bloc fonctionnel bistable (Reset dominant)</td></tr>
<tr><td>Forme graphique
 
</td><td>Corps du bloc fonctionnel
 
</td></tr>
</table>

Tableau 2 Blocs fonctionnels bistables (d'après la norme [IEC 61131-3])

Dans cette définition informelle, la valeur Q_1 de la sortie à un instant donné est construite implicitement à partir de la valeur des entrées au même instant et la valeur de Q_1 à l'instant précédent. Nous sommes en présence d'une définition sous forme d'équations récurrentes. Si cette définition par récurrence est couramment utilisée, elle n'est cependant pas forcément nécessaire lorsqu'il est possible de faire référence aux valeurs antérieures des entrées. Il suffit de remplacer la valeur de Q_1 à l'instant précédent par l'information qu'elle représente, c'est-à-dire l'existence d'une date antérieure pour laquelle la mémoire a été mise à un et depuis laquelle la mémoire n'a jamais été remise à zéro.

A.3 Opérations permettant de décrire un comportement temporisé

La norme [IEC 61131-3], norme relative aux langages de programmation des API, propose deux blocs fonctionnels qui décrivent des comportements temporisés. La définition proposée dans cette norme est la suivante (tableau 3) :

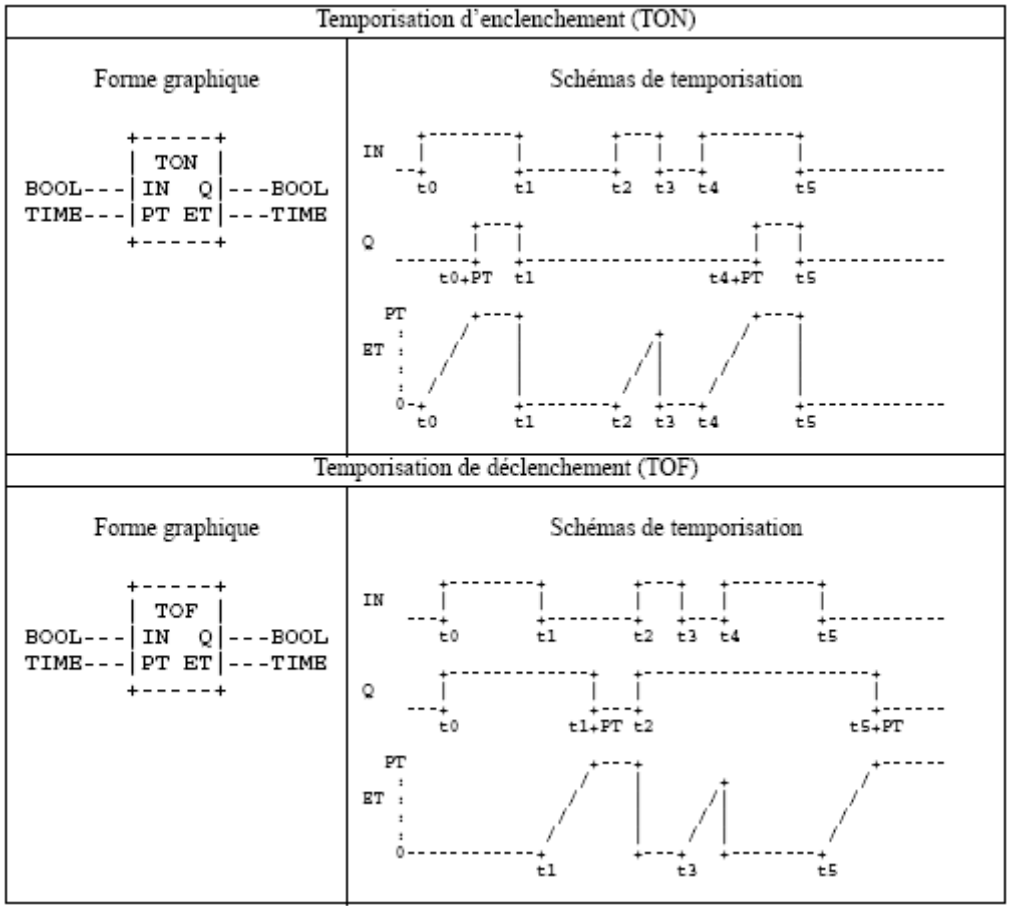


Tableau 3 Blocs fonctionnels temporisateurs (d'après la norme [IEC 61131-3])

Dans cette norme, le comportement de ces blocs fonctionnels est présenté sous forme de chronogrammes décrivant le comportement de la sortie Q en fonction de l'entrée IN. La variable ET est une variable «continue» qui croit linéairement avec le temps jusqu'à la valeur PT. Cette croissance s'effectue pour le bloc TON lorsque l'entrée IN est présente, et pour le bloc TOF, lorsque l'entrée IN est absente.

B. Démonstrations du chapitre 2 : [2.10] à [2.26]

Démonstration de l'équation [2.10] : $f \cdot f = f$

$$\forall t \in \mathbb{R}^{+*}, \quad (f \cdot f)(t) = f(t) \wedge f(t) = f(t)$$

donc, $f \cdot f = f$ □

Démonstration de l'équation [2.11] : $f + f = f$

$$\forall t \in \mathbb{R}^{+*}, \quad (f + f)(t) = f(t) \vee f(t) = f(t)$$

donc, $f + f = f$ □

Démonstration de l'équation [2.12] : $f \cdot 0* = 0*$

$$\forall t \in \mathbb{R}^{+*}, \quad (f \cdot 0*)(t) = f(t) \wedge 0*(t) = f(t) \wedge 0 = 0 = 0*(t)$$

donc, $f \cdot 0* = 0*$ □

Démonstration de l'équation [2.13] : $f + 1* = 1*$

$$\forall t \in \mathbb{R}^{+*}, \quad (f + 1)(t) = f(t) \vee 1*(t) = f(t) \vee 1 = 1 = 1*(t)$$

donc, $f + 1* = 1*$ □

Démonstration de l'équation [2.14] : $f + (f \cdot g) = f$

en utilisant [2.5], [2.3], [2.13] et [2.5]

$$f + (f \cdot g) = (f \cdot 1*) + (f \cdot g) = f \cdot (1* + g) = f \cdot 1*$$

donc, $f + (f \cdot g) = f$ □

Démonstration de l'équation [2.15] : $f \cdot (f + g) = f$

en utilisant [2.3], [2.10] et [2.14]

$$f \cdot (f + g) = (f \cdot f) + (f \cdot g) = f + (f \cdot g)$$

donc, $f + (f \cdot g) = f$ □

Démonstration de l'équation [2.16] : $f \cdot (g \cdot h) = (f \cdot g) \cdot h$

$$\begin{aligned} \forall t \in \mathbb{R}^{+*}, \quad f \cdot (g \cdot h)(t) &= f(t) \wedge (g \cdot h)(t) \\ &= f(t) \wedge \Big(g(t) \wedge h(t) \Big) \\ &= f(t) \wedge g(t) \wedge h(t) \\ &= \Big(f(t) \wedge g(t) \Big) \wedge h(t) \\ &= (f \cdot g)(t) \wedge h(t) \\ &= \Big((f \cdot g) \cdot h \Big)(t) \end{aligned}$$

donc, $f \cdot (g \cdot h) = (f \cdot g) \cdot h$ □

Démonstration de l'équation [2.17] : $f + (g + h) = (f + g) + h$

$$\begin{aligned} \forall t \in \mathbb{R}^{+*}, \quad \Big(f + (g + h) \Big)(t) &= f(t) \vee (g + h)(t) \\ &= f(t) \vee \Big(g(t) \vee h(t) \Big) \\ &= f(t) \vee g(t) \vee h(t) \\ &= \Big(f(t) \vee g(t) \Big) \vee h(t) \\ &= (f + g)(t) \vee h(t) \\ &= \Big((f + g) + h \Big)(t) \end{aligned}$$

donc, $f + (g + h) = (f + g) + h$ □

Démonstration de l'équation [2.18] : $\overline{\overline{f}} = f$

$$\begin{aligned} \forall t \in \mathbb{R}^{+*}, \quad \overline{\overline{f}}(t) &= \neg \overline{f}(t) \\ &= \neg \Big(\neg f(t) \Big) \\ &= \neg \neg f(t) \\ &= f(t) \end{aligned}$$

donc, $\overline{\overline{f}} = f$ □

Démonstration de l'équation [2.19] : $\overline{(f \cdot g)} = \overline{f} + \overline{g}$

$$\begin{aligned} \forall t \in \mathbb{R}^{+*}, \quad \overline{(f \cdot g)}(t) &= \neg (f \cdot g)(t) \\ &= \neg \Big(f(t) \wedge (g)(t) \Big) \\ &= \neg f(t) \vee \neg g(t) \\ &= \overline{f}(t) \vee \overline{g}(t) \\ &= (\overline{f} + \overline{g})(t) \end{aligned}$$

donc, $\overline{(f \cdot g)} = \overline{f} + \overline{g}$ □

Démonstration de l'équation [2.20] : $\overline{(f + g)} = \overline{f} \cdot \overline{g}$

$$\forall t \in \mathbb{R}^{+*}, \quad \overline{(f + g)}(t) = \neg (f + g)(t)$$

$$\begin{aligned} &= \neg\Big(f(t) \vee (g)(t)\Big) \\ &= \neg f(t) \wedge \neg g(t) \\ &= \overline{f}(t) \wedge \overline{g}(t) \\ &= (\overline{f} \cdot \overline{g})(t) \end{aligned}$$

donc, $\overline{(f+g)} = \overline{f} \cdot \overline{g}$ □

Démonstration de l'équation [2.21] : $f + (\overline{f} \cdot g) = f + g$

en utilisant [2.14], [2.17], [2.3], [2.8] et [2.5],

$$\begin{aligned} f + (\overline{f} \cdot g) &= \Big(f + (f \cdot g)\Big) + (\overline{f} \cdot g) \\ &= f + \Big((f \cdot g) + (\overline{f} \cdot g)\Big) \\ &= f + \Big((f + \overline{f}) \cdot g)\Big) \\ &= f + (1* \cdot g) \\ &= f + g \end{aligned}$$ □

Démonstration de l'équation [2.22] : $f \cdot g + \overline{f} \cdot h + g \cdot h = f \cdot g + \overline{f} \cdot h$

en utilisant [2.5], [2.8], [2.3], [2.17], [2.1], [2.17] et [2.14]

$$\begin{aligned} f \cdot g + \overline{f} \cdot h + g \cdot h &= f \cdot g + \overline{f} \cdot h + \Big((1*) \cdot (g \cdot h)\Big) \\ &= f \cdot g + \overline{f} \cdot h + \Big((f + \overline{f}) \cdot (g \cdot h)\Big) \\ &= f \cdot g + \overline{f} \cdot h + (f \cdot g \cdot h) + (\overline{f} \cdot g \cdot h) \\ &= \Big(f \cdot g + (f \cdot g \cdot h)\Big) + \Big(\overline{f} \cdot h + (\overline{f} \cdot g \cdot h)\Big) \\ &= f \cdot g + \overline{f} \cdot h \end{aligned}$$ □

Démonstration de l'équation [2.23] : $\overline{1*} = 0*$

$\forall t \in \mathbb{R}^{+*}$,

$$\begin{aligned} (\overline{1*})(t) &= \neg\Big((1*)(t)\Big) \\ &= \neg 1 \\ &= 0 \\ &= 0*(t) \end{aligned}$$

donc, $(\overline{1*}) = 0*$ □

Démonstration de l'équation [2.24] : $\overline{0*} = 1*$

$\forall t \in \mathbb{R}^{+*}$,

$$\begin{aligned} (\overline{0*})(t) &= \neg\Big((0*)(t)\Big) \\ &= \neg 0 \\ &= 1 \\ &= 1*(t) \end{aligned}$$

donc, $(\overline{0*}) = 1*$ □

Démonstration de l'équation [2.25] : $f + g = 0* \quad \Leftrightarrow \quad \begin{cases} f = 0* \\ g = 0* \end{cases}$

$$f + g = 0* \quad \Leftrightarrow \quad \forall t \in \mathbb{R}^{+*}, \quad \left(f(t) \vee g(t) = 0 \right)$$

$$\Leftrightarrow \quad \begin{cases} \forall t \in \mathbb{R}^{+*}, & \left(f(t) = 0 \right) \\ \forall t \in \mathbb{R}^{+*}, & \left(g(t) = 0 \right) \end{cases}$$

$$donc, \quad f + g = 0* \quad \Leftrightarrow \quad \begin{cases} f = 0* \\ g = 0* \end{cases}$$

□

Démonstration de l'équation [2.26] : $f \cdot g = 1* \quad \Leftrightarrow \quad \begin{cases} f = 1* \\ g = 1* \end{cases}$

$$f \cdot g = 1* \quad \Leftrightarrow \quad \forall t \in \mathbb{R}^{+*}, \quad \left(f(t) \wedge g(t) = 1 \right)$$

$$\Leftrightarrow \quad \begin{cases} \forall t \in \mathbb{R}^{+*}, & \left(f(t) = 1 \right) \\ \forall t \in \mathbb{R}^{+*}, & \left(g(t) = 1 \right) \end{cases}$$

$$donc, \quad f \cdot g = 1* \quad \Leftrightarrow \quad \begin{cases} f = 1* \\ g = 1* \end{cases}$$

□

C. Représentation graphique des spécifications sur $\mathbb{I}$

Cette section n'a pour but que d'introduire les spécifications sur $\mathbb{I}$, aux non experts dans cette algèbre, à l'aide d'une représentation graphique. Nous précisons cependant que ces représentations ne sont pas des automates à états dans le sens strict du terme. En effet, des spécifications sur $\mathbb{I}$, tel que l'élément 0*, n'ont pas un modèle strictement équivalent en automate à états, modèle couramment utilisé par les automaticiens. Pour cela, nous nous appuierons sur l'exemple 3 (défini au chapitre IV).

1) La spécification de type **SF-T1** considère premièrement, un changement d'état de la sortie et deuxièmement, l'unité-stabilité.

Par exemple, l'assertion A2 de l'exemple 3 (voir la figure 46a) : $P\_prise \cdot P\_basse \le Aspirer$, demande d'aspirer si l'aspiration n'était pas activée quand $P\_prise \cdot P\_basse$ vaut 1. En effet, $(P\_prise \cdot P\_basse) \cdot \overline{Aspirer} \le Aspirer \Leftrightarrow P\_prise \cdot P\_basse \le Aspirer$ (en utilisant [2.51]). Cette assertion demande en plus, pour garantir l'unité-stabilité, de continuer l'aspiration tant que $P\_prise \cdot P\_basse$ vaut 1.

2) La spécification de type **SF-T2** : $F_1(u,X) \le F_2(X)$, qui peut être réécrite comme $F_1(u,X) \cdot \overline{F_2(X)} \le F_2(X)$ en utilisant [2.51], considère un changement d'état vers l'ensemble des états d'arrivée $F_2(X)$. L'expression $F_1(u,X) \cdot \overline{F_2(X)}$ nous indique les états de départ et la condition de transition. Nous montrons la représentation de cette spécification dans la figure c1, la représentation de : $f \cdot X_1 \le X_2$ dans la figure 46c2 et, en plus, l'assertion A5 de l'exemple 3 dans la figure 46c3.

Notons que l'assertion A5 de l'exemple 3 : $P\_depose \cdot P\_haute \le \overline{Avancer} \cdot Descendre$, est un cas particulier car elle n'a qu'un état d'arrivée. D'après [2.51], elle peut être réécrite comme :

$$P\_depose \cdot P\_haute \cdot (Avancer + \overline{Descendre}) \le \overline{Avancer} \cdot Descendre$$

L'assertion A5 sollicite de descendre et de ne pas avancer le préhenseur quand $P\_prise \cdot P\_basse$ vaut 1. C'est-à-dire, A5 demande de transiter ver l'état d'arrivée : $\overline{Avancer} \cdot Descendre$ et d'y rester, depuis tous les états $Avancer + \overline{Descendre}$ quand $P\_depose \cdot P\_haute$ vaut 1, tel qu'il montré dans la figure 46c.

3) La spécification de type **SF-T3** traduise des comportements observés par les détecteurs en mode normal de fonctionnement, comme par exemple la non-simultanéité des valeurs vraies de deux détecteurs fin de course, tel que l'assertion A9 de l'exemple 3 (voir la figure 46b). La modélisation de ce contrainte est fait à l'aide de l'élément 1*.

4) La spécification de type **SF-T4** (figure 46e1) traduise des contraintes sur les signaux de sortie représentant des états interdits en utilisant l'élément 1*. Une modélisation plus simple de cette contrainte est montrée dans la figure 46e2, en utilisant l'assertion A14 de l'exemple 3.

5) La spécification de type **SF-T5** tel que $u_1 \le REP_X(f,g)$, traduise le désire de rester dans l'état X suit à l'entrée u_1 tel que $u_1 \in \overline{f} \cdot \overline{g}$ (voir figure 46d1). C'est-à-dire, u_1 n'appartient ni aux transitions d'arrivée ni aux transitions de départ de l'état X. Une flèche qui reste dans cet état est suffisante pour représenter la SF-T5.

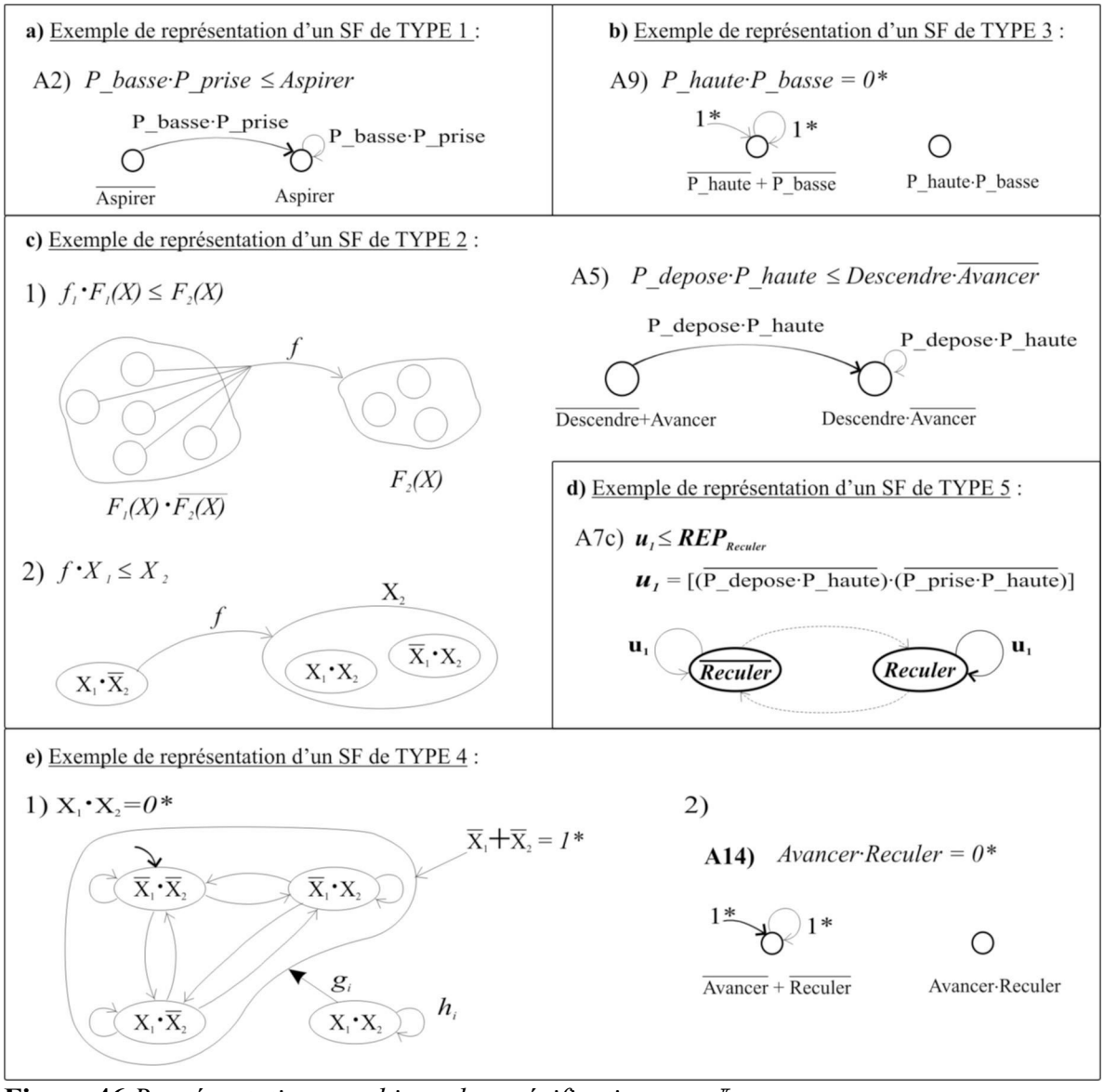


Figure 46 *Représentation graphique des spécifications sur 𝕀*

D. Démonstrations

D.1 Démonstrations relatifs aux prédicats k_3, k_{4a} et k_{4b}

Analysons la validité de ces prédicats $\forall f, g$ tels que $f \cdot g = 0*$ du système 7.

Nous allons nous appuyer sur les deux points suivants pour faire la validation :

1) La contrainte de l'existence de une solution : $f \cdot g = 0*$ est imposée par la cohérence. Cette expression, qui doit être toujours accomplie, est équivalente à $\overline{f} + \overline{g} = 1*$ (en utilisant [2.54] ⇔ [2.56]).
2) Il est établit dans le troisième cas de figure que $\overline{f} \cdot \overline{g} \neq 0*$. En effet, si $\overline{f} \cdot \overline{g} = 0*$ alors nous serions dans le deuxième cas de figure. De ce fait, nous pouvons affirmer que pour le troisième cas de figure, $\exists t : \left[(\overline{f} \cdot \overline{g})(t) = 1 \right]$ à partir de la définition de la relation de différence (2.3.1).

D'une part, nous savons depuis le point 1 que $\overline{f} + \overline{g} = 1*$, l'expression suivante est donc toujours vraie

$$1(t) = \begin{cases} 1(t) & \forall t < t_1 \\ \left[\exists t_1 : \left((f+g)(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\overline{f} + \overline{g})(d) = 1 \right) \right] & \forall t \geq t_1 \end{cases}$$

Le deuxième terme existe $\forall t \geq t_1$ et elle est égale à 1. De plus, comme $f \cdot g = 0*$ alors $f + g = f \cdot \overline{g} + \overline{f} \cdot g$ et par conséquent les deux derniers termes sont donc disjoints. Ainsi, comme la deuxième expression est valide alors celle-ci, divisé en deux espaces de temps comme suit, est aussi valide :

$$1(t) = \begin{cases} \left[\exists t_1 : \left((f \cdot \overline{g})(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\overline{f} + \overline{g})(d) = 1 \right) \right] & \forall t > t_1 \\ \left[\exists t_1 : \left((\overline{f} \cdot g)(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\overline{f} + \overline{g})(d) = 1 \right) \right] & \forall t > t_1 \end{cases}$$

Comme $f \cdot g = 0*$ nous savons d'une part que $f = f \cdot \overline{g}$ et $g = \overline{f} \cdot g$ d'après [2.37] ⇔ [2.35]

et [2.37] ⇔ [2.38] respectivement. De ce fait et en séparant le terme $\overline{f}$ en $\overline{g}$ nous avons

$$= \begin{cases} \left[\exists t_1 : \left(f(t_1)=1\right) \wedge \left(\forall d \in]t_1, t], (\overline{f})(d)=1\right)\right] & \forall t > t_1 \\ \left[\exists t_1 : \left(f(t_1)=1\right) \wedge \left(\forall d \in]t_1, t], (\overline{g})(d)=1\right)\right] & \forall t > t_1 \\ \left[\exists t_1 : \left(g(t_1)=1\right) \wedge \left(\forall d \in]t_1, t], (\overline{f})(d)=1\right)\right] & \forall t > t_1 \\ \left[\exists t_1 : \left(g(t_1)=1\right) \wedge \left(\forall d \in]t_1, t], (\overline{g})(d)=1\right)\right] & \forall t > t_1 \end{cases}$$

Comme la dernière expression est valide alors le deuxième et le troisième terme sont aussi valides. Ces termes sont égaux aux k_3 et k_{4a} respectivement alors

les prédicats k_3 (figure 30c) et k_{4a} (figure 30d) sont donc valides . □

D'autre part, le point 2 indique que $\left[(\overline{f} \cdot \overline{g})(t)=1\right]$ existe, l'expression suivante est donc valide :

$$(\overline{f} \cdot \overline{g})(t) \quad = \quad \begin{cases} \left[\forall d \in]0, t] : (\overline{f} \cdot \overline{g})(d)=1\right] & \forall t < t_1 \\ \left[\exists t_1 \le t : \left((f \vee g)(t_1)=1\right) \wedge (\overline{f} \cdot \overline{g})(t)\right] & \forall t \ge t_1 \end{cases}$$

Cette expression représente $(\overline{f} \cdot \overline{g})(t)$ divisé en deux espaces de temps. La première partie de l'expression est égale au prédicat k_{4b}. Comme la dernière expression est valide,

le prédicat k_{4b} (figure 30e) est donc valide □

D.2 Démonstrations de l'hypothèse hyp_2 de la solution avec mémoire

Nous allons vérifier ci-dessous pour la solution particulière nommée avec mémoire (voir 5.5), que les contraintes données par hyp_2 sont bien respectées : $k_3 \cdot k_4 + f \cdot k_4 + g \cdot k_3 + \overline{f} \cdot \overline{g} \cdot \overline{k_3} \cdot \overline{k_4} = 0$ *.

Nous procéderons en faisant la démonstration de chaque terme car [2.25] :

$$\left(hyp_2\right)\quad k_3\cdot k_4 + f\cdot k_4 + g\cdot k_3 + \overline{f}\cdot\overline{g}\cdot\overline{k_3}\cdot\overline{k_4} = 0* \qquad \Leftrightarrow \qquad \begin{cases} hyp_{2a}) & k_3\cdot k_4 = 0* \\ hyp_{2b}) & f\cdot k_4 = 0* \\ hyp_{2c}) & g\cdot k_3 = 0* \\ hyp_{2d}) & \overline{f}\cdot\overline{g}\cdot\overline{k_3}\cdot\overline{k_4} = 0* \end{cases}$$

Au cours des démonstrations suivantes, les instants $t, t_1 \in \mathbb{R}^{+*}$.

Démonstration de l'hypothèse 2a : $k_3\cdot k_4 = 0*$

Nous pouvons constater que les prédicats k_3, k_{4a} et k_{4b},sont disjoints par une condition dans le passé :

1) $k_3 \wedge k_{4a} = 0*$, car le prédicat $\left[\exists t_1 < t : \left(f(t_1) = 1\right)\right] \wedge \left[\exists t_1 < t : \left(g(t_1) = 1\right)\right]$ est toujours faux, du fait que $f\cdot g = 0*$ est une condition nécessaire pour avoir une solution.
2) $k_3 \wedge k_{4b} = 0*$, car k_{4b} est vrai si et seulement si le prédicat $\left[\exists t_1 < t : f(t_1) = 1\right]$ est faux. Or, si ce prédicat est faux alors k_3 est aussi faux. Les prédicats k_3 et k_{4b} sont donc disjoints.
3) $k_{4a} \wedge k_{4b} = 0*$, car k_{4b} est vrai ssi le prédicat $\left[\exists t_1 < t : g(t_1) = 1\right]$ est faux. Or, s'il est faux alors k_{4a} est aussi faux. Les prédicats k_{4a} et k_{4b} sont donc disjoints.

En substituant les valeurs de k_3, k_4 et les points 1 et 2 nous avons

$$k_3\cdot k_4 = k_3\cdot(k_{4a} + k_{4b}) = k_3\cdot k_{4a} + k_3\cdot k_{4b} = 0* + 0* = 0*$$ □

Démonstration de l'hypothèse 2b : $f\cdot k_4 = 0*$

Nous devons démontrer alors que $f\cdot k_4 = f\cdot(k_{4a} + k_{4b}) = 0*$:

$\forall t \in \mathbb{R}^{+*}$,

$$f(t) \wedge \left(k_{4a}(t) \vee k_{4b}(t)\right) = f(t) \wedge \left(\begin{array}{c} \left[\exists t_1 : \left(g(t_1) = 1\right) \wedge \left(\forall d \in]t_1, t], (\overline{f})(d) = 1\right)\right] \\ \vee \left[\forall d \in]0, t], (\overline{f}\cdot\overline{g})(d) = 1\right] \end{array}\right)$$

$$= f(t) \wedge \left(\begin{array}{c} \neg f(t) \wedge \left[\exists t_1 : \left(g(t_1) = 1\right) \wedge \left(\forall d \in]t_1, t[, (\overline{f})(d) = 1\right)\right] \\ \vee \neg f(t) \wedge \neg g(t) \left[\forall d \in]0, t[, (\overline{f}\cdot\overline{g})(d) = 1\right] \end{array}\right)$$

Nous avons donc $\forall t \in \mathbb{R}^{+*}, f(t) \wedge (k_{4a} + k_{4b})(t) = 0(t) \vee 0(t) = 0(t)$, alors

$f \cdot (k_{4a} + k_{4b}) = 0*$ □

Démonstration de l'hypothèse 2c : $g \cdot k_3 = 0*$

Nous devons démontrer alors que $g \cdot k_3 = 0*$:

$\forall t \in \mathbb{R}^{+*}$,

$$g(t) \wedge k_3(t) = g(t) \wedge \left[\exists t_1 < t : \left(\left(f(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\overline{g})(d) = 1 \right) \right]$$

$$= g(t) \wedge \left(\neg g(t) \wedge \left[\exists t_1 < t : \left(\left(f(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t[, (\overline{g})(d) = 1 \right) \right) \right] \right)$$

Nous avons donc $\forall t \in \mathbb{R}^{+*}, g(t) \wedge k_3(t) = 0(t)$, alors

$g \cdot k_3 = 0*$ □

Démonstration de l'hypothèse 2d : $\overline{f} \cdot \overline{g} \cdot \overline{k_3} \cdot \overline{k_4} = 0*$

Nous allons faire la démonstration de l'hypothèse 2d en utilisant [2.37] $\Leftrightarrow$ [2.35]) :
$\overline{f} \cdot \overline{g} \cdot \overline{k_3} \cdot \overline{k_4} = 0* \Leftrightarrow \overline{f} \cdot \overline{g} = \overline{f} \cdot \overline{g} \cdot (k_3 + k_4)$. Nous devons démontrer alors que $\overline{f} \cdot \overline{g} = \overline{f} \cdot \overline{g} \cdot (k_3 + k_{4a} + k_{4b})$:

$\forall t \in \mathbb{R}^{+*}$,

$$(\overline{f} \cdot \overline{g})(t) \wedge (k_3 + k_{4a} + k_{4b})(t) = (\overline{f} \cdot \overline{g})(t) \wedge \left(k_3(t) \vee k_{4a}(t) \vee k_{4b}(t) \right)$$

$$= (\overline{f} \cdot \overline{g})(t) \wedge \left(\begin{array}{l} \left[\exists t_1 : \left(f(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\overline{g})(d) = 1 \right) \right] \vee \\ \left[\exists t_1 : \left(g(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\overline{f})(d) = 1 \right) \right] \vee \left[\forall d \in]0, t], (\overline{f} \cdot \overline{g})(d) = 1 \right] \end{array} \right)$$

$$= (\overline{f} \cdot \overline{g})(t) \wedge \left(\begin{array}{l} \left[\exists t_1 : \left(f(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\overline{f} \cdot \overline{g})(d) = 1 \right) \right] \vee \\ \left[\exists t_1 : \left(g(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t], (\overline{f} \cdot \overline{g})(d) = 1 \right) \right] \vee \left[\forall d \in]0, t], (\overline{f} \cdot \overline{g})(d) = 1 \right] \end{array} \right)$$

$$= (\overline{f} \cdot \overline{g})(t) \wedge \left(\begin{array}{l} \left[\exists t_1 : \left(f(t_1) = 1 \right) \wedge \left((\overline{f} \cdot \overline{g})(t) = 1 \right) \right] \\ \vee \left[\exists t_1 : \left(g(t_1) = 1 \right) \wedge \left((\overline{f} \cdot \overline{g})(t) = 1 \right) \right] \vee \left[\forall d \in]0, t], (\overline{f} \cdot \overline{g})(d) = 1 \right] \end{array} \right)$$

$$= (\overline{f} \cdot \overline{g})(t) \wedge \left(\left[\exists t_1 : \left(f(t_1) \vee g(t_1) = 1 \right) \wedge \left((\overline{f} \cdot \overline{g})(t) = 1 \right) \right] \vee \left[\forall d \in]0, t], (\overline{f} \cdot \overline{g})(d) = 1 \right] \right)$$
$$= (\overline{f} \cdot \overline{g})(t)$$

Nous avons donc $\forall t \in \mathbb{R}^{+*}, (\overline{f} \cdot \overline{g})(t) \wedge (k_3 + k_{4a} + k_{4b})(t) = (\overline{f} \cdot \overline{g})(t)$, alors

$$(\overline{f} \cdot \overline{g}) \cdot (k_3 + k_{4a} + k_{4b}) = (\overline{f} \cdot \overline{g})$$ □

D.3 Démonstrations des théorèmes relatifs aux opérations SR et RS

Nous avons présenté (voir 5.5.2) la démonstration formelle de la résolution du cas général d'un système d'équations logiques sur $\mathbb{I}$, donné par le système 8 du chapitre 5. Ainsi, dans cette section nous ferons les démonstrations des théorèmes [5.1] à [5.4] et [3.13] à [3.40] concernant les opérations SR et RS.

Nous soulignons que par la suite nous n'allons pas représenter ni hyp_1 ni hyp_2, car dans nos travaux nous ne considérons que la solution mémoire (développée au chapitre 5) et pour ce type de solution hyp_1 et hyp_2 sont toujours vraies (voir 5.5).

Nous pouvons exprimer la solution d'un système d'équations par les opérations SR, RS et REP comme il est indiqué dans le théorème [5.1] et démontré ci-dessous.

Démonstration du théorème [5.1] :

D'après la solution temporelle donnée par le système 8 du chapitre 5, le système du [5.1] a la solution temporelle suivante :

$$\forall t \in \mathbb{R}^{+*},$$
$$X(t) = (s + k_1)(t) \vee \left[\exists t_1 < t : \left(\left((s + k_1)(t_1) = 1 \right) \wedge \left(\forall d \in]t_1, t[, \left((\overline{r + k_2})(d) = 1 \right) \right) \right) \right],$$

ce théorème sera démontré en deux temps :

1) Comme le système 8 est cohérent alors $(s + k_1) \cdot (r + k_2) = 0$*. En utilisant [2.37] ⇔ [2.39], nous avons $(s + k_1) + (\overline{r + k_2}) = \overline{r + k_2}$. Par conséquent,

$$\forall t \in \mathbb{R}^{+*},$$
$$X(t) = (s + k_1)(t) \vee \left[\exists t_1 < t : \left(\begin{array}{l} \left((s + k_1)(t_1) = 1 \right) \\ \quad \wedge \left(\forall d \in]t_1, t[, \left(\left((s + k_1) + \overline{r + k_2} \right)(d) = 1 \right) \right) \end{array} \right) \right]$$

2) D'autre part, comme le système 8 est cohérent alors $(s+k_1)\cdot(r+k_2)=0*$, en utilisant [2.37] $\Leftrightarrow$ [2.35], nous avons $(s+k_1)\cdot(\overline{r+k_2})=(s+k_1)$. Par conséquent,

$\forall t\in\mathbb{R}^{+*}$,

$$X(t)=\Big((s+k_1)\cdot\overline{r+k_2}\Big)(t)\vee\left[\exists t_1<t:\left(\begin{array}{r}\Big(\Big((s+k_1)\cdot(\overline{r+k_2})\Big)(t_1)=1\Big)\\ \wedge\Big(\forall d\in]t_1,t],(\overline{r+k_2})(d)=1\Big)\end{array}\right)\right]$$

$$X(t)=\Big((s+k_1)\cdot\overline{r+k_2}\Big)(t)\vee\left[\exists t_1<t:\left(\begin{array}{r}\Big((s+k_1)(t_1)=1\Big)\\ \wedge\Big(\forall d\in[t_1,t],(\overline{r+k_2})(d)=1\Big)\end{array}\right)\right]$$

$$X(t)=\Big((s+k_1)(t)\wedge(\overline{r+k_2})(t)\Big)\vee\left[\exists t_1<t:\left(\begin{array}{r}\Big((s+k_1)(t_1)=1\Big)\\ \wedge\Big(\forall d\in[t_1,t],(\overline{r+k_2})(d)=0\Big)\end{array}\right)\right]$$

Ainsi, en considérant les signaux $s+k_1$ et $r+k_2$, il s'avère que les points 1 et 2 sont égaux à la **définition 9** et **10**, c'est-à-dire, $SR(s+k_1,r+k_2)$ et $RS(s+k_1,r+k_2)$, respectivement □

Démonstration du théorème [5.2] **:**

Ce théorème sera démontre à partir du cas général donné par [5.1]. En effet, en considérant $k_s=k_r=k_h=h=0*$, d'après sa solution donnée par SR et RS ([5.1]) nous avons en utilisant les théorèmes suivantes [2.24], [2.5], [2.6], [2.12] et [2.31]

$$\left\{\begin{array}{ll}\left\{\begin{array}{ll}1a) & s\le X\\ 2a) & r\le\overline{X}\\ 3a) & \bar{s}\cdot\bar{r}\le REP\end{array}\right.\\ \left\{\begin{array}{ll}1a) & 0*\le X\\ 2a) & 0*\le\overline{X}\\ 3a) & 0*\le REP\end{array}\right.\\ \left\{\begin{array}{ll}hyp_1) & s\cdot r+s\cdot(\bar{s}\cdot\bar{r})+r\cdot(\bar{s}\cdot\bar{r})=0*\\ hyp_2) & 0*\cdot0*+0*\cdot0*+0*\cdot0*=0*\\ hyp_3) & \left(\begin{array}{r}s\cdot(0*+0*)+r\cdot(0*+0*)\\ +(\bar{s}\cdot\bar{r})\cdot(0*+0*)\end{array}\right)=0*\\ hyp_4) & \bar{s}\cdot\bar{r}\cdot\overline{\Big(\bar{s}\cdot\bar{r}\Big)}\cdot\overline{0*}\cdot\overline{0*}\cdot\overline{0*}=0*\end{array}\right.\end{array}\right.\Leftrightarrow\left\{\begin{array}{l}s\le X\\ r\le\overline{X}\\ \bar{s}\cdot\bar{r}\le REP\\ s\cdot r=0*\end{array}\right.\Leftrightarrow\begin{array}{c}X=SR(s,r)\\ \text{où bien}\\ X=RS(s,r)\end{array}$$

Nous allons démontrer les théorèmes [5.3] et [5.4]. Ces théorèmes montrent les systèmes d'équations qui ont comme solution les opérations SR et RS, respectivement.

Démonstration l'équivalence [5.3] **:**

La solution avec mémoire du $SR(s,r)$ donné par [5.2] est vrai si la cohérence $(s \cdot r = 0^*)$ est respectée. Cependant si $s \cdot r = 0^*$ alors $r = \bar{s} \cdot r$ ([2.37] $\Leftrightarrow$ [2.38]). Le théorème [5.2] implique, en utilisant [2.19], [2.18] et [2.7]

$$\begin{cases} 1a) & s \le X \\ 2a) & (\bar{s} \cdot r) \le \overline{X} \\ 3a \cdot hyp_{2a}) & \bar{s} \cdot \overline{(\bar{s} \cdot r)} \le REP(X) \\ coh\acute{e}rence) & s \cdot (\bar{s} \cdot r) = 0^* \end{cases} \Leftrightarrow \begin{cases} 1a) & s \le X \\ 2a) & \bar{s} \cdot r \le \overline{X} \\ 3a \cdot hyp_{2a}) & \bar{s} \cdot (s + \bar{r}) \le REP(X) \\ coh\acute{e}rence) & 0^* = 0^* \end{cases}$$

nous avons donc à partir de [5.2] (en utilisant [2.3], [2.7] et [2.6])

$$\begin{cases} 1a) & s \le X \\ 2a) & \bar{s} \cdot r \le \overline{X} \\ 3a) & \bar{s} \cdot \bar{r} \le REP(X) \end{cases} \Leftrightarrow X = SR(s,r) \qquad \square$$

Démonstration l'équivalence [5.4] **:**

La solution avec mémoire du $RS(s,r)$ donnée par [5.2] est vraie si la cohérence $(s \cdot r = 0^*)$ est respectée. Cependant si $s \cdot r = 0^*$ alors $s = s \cdot \bar{r}$ ([2.37] $\Leftrightarrow$ [2.38]). Le théorème [5.2] implique, en utilisant [2.19], [2.18] et [2.7]

$$\begin{cases} 1a) & s \cdot \bar{r} \le X \\ 2a) & r \le \overline{X} \\ 3a \cdot hyp_{2a}) & \overline{(s \cdot \bar{r})} \cdot \bar{r} \le REP(X) \\ coh\acute{e}rence) & (s \cdot \bar{r}) \cdot r = 0^* \end{cases} \Leftrightarrow \begin{cases} 1a) & s \cdot \bar{r} \le X \\ 2a) & r \le \overline{X} \\ 3a \cdot hyp_{2a}) & (\bar{s} \cdot + r) \cdot \bar{r} \le REP(X) \\ coh\acute{e}rence) & 0^* = 0^* \end{cases}$$

nous avons donc à partir de [5.2] (en utilisant [2.3], [2.7] et [2.6])

$$\begin{cases} 1a) & s \cdot \bar{r} \le X \\ 2a) & r \le \overline{X} \\ 3a) & \bar{s} \cdot \bar{r} \le REP(X) \end{cases} \Leftrightarrow X = RS(s,r) \qquad \square$$

Remarques :

Le Théorème [5.3] n'a pas comme solution $RS(s,r)$ que si $s \cdot r = 0^*$ (voir [5.2]).

Le Théorème [5.4] n'a pas comme solution $SR(s,r)$ que si $s \cdot r = 0^*$ (voir [5.2]).

A partir de la définition mathématique des opérations SR et RS, il est possible de démontrer les théorèmes [3.13] à [3.40]. Nous allons utiliser cependant les théorèmes [5.3] et [5.4] car ils serviront à rendre plus aisés les démonstrations de ces théorèmes.

Démonstration de la relation [3.13] : $s \le SR(s,r)$

Ce théorème sera démontré en utilisant [5.3]. En effet, comme la solution avec mémoire est $X = SR(s,r)$, alors depuis le point 1a) de [5.3] nous avons,

$s \le SR(s,r)$ □

Démonstration de la relation [3.14] : $\bar{s} \cdot r \le \overline{SR(s,r)}$

Ce théorème sera démontré en utilisant [5.3]. En effet, si la solution est $X = SR(s,r)$, alors $\overline{X} = \overline{SR(s,r)}$ ([2.54] $\Leftrightarrow$ [2.56]) et depuis le point 2a) de [5.3] nous avons,

$\bar{s} \cdot r \le \overline{SR(s,r)}$ □

Démonstration de l'équation [3.15] : $SR(s,r) = SR(s,\bar{s} \cdot r)$

Ce théorème sera démontré en utilisant [5.3]. Ainsi, $X = SR(s,\bar{s} \cdot r)$ équivaut au système suivant. Puis en utilisant [2.16], [2.10], [2.19], [2.18] et [2.3], [2.7], [2.6] nous avons,

$$\begin{cases} 1a) & s \le X \\ 2a) & \bar{s} \cdot (\bar{s} \cdot r) \le \overline{X} \\ 3a) & \bar{s} \cdot \overline{(\bar{s} \cdot r)} \le REP(X) \end{cases} = \begin{cases} 1a) & s \le X \\ 2a) & \bar{s} \cdot r \le \overline{X} \\ 3a) & \bar{s} \cdot (s + \bar{r}) \le REP(X) \end{cases} = \begin{cases} 1a) & s \le X \\ 2a) & \bar{s} \cdot r \le \overline{X} \\ 3a) & \bar{s} \cdot \bar{r} \le REP(X) \end{cases}$$

et en utilisant [5.3], nous avons $X = SR(s,r)$. Par transitivité de la relation d'égalité, $SR(s,\bar{s} \cdot r) = SR(s,r)$ □

Démonstration de l'équation [3.16] : $SR(s,1^*) = s$

Ce théorème sera démontré en utilisant [5.3]. Ainsi, $X = SR(s,1^*)$ équivaut au système suivant. Puis en utilisant [2.5], [2.23], [2.12]; [2.31]; [2.5] nous avons,

$$\begin{cases} 1a) & s \le X \\ 2a) & \bar{s} \cdot 1^* \le \overline{X} \\ 3a) & \bar{s} \cdot \overline{1^*} \le REP(X) \end{cases} = \begin{cases} 1a) & s \le X \\ 2a) & \bar{s} \le \overline{X} \\ 3a) & 0^* \le REP(X) \end{cases} = \begin{cases} 1a) & s \le X \\ 2a) & \bar{s} \le \overline{X} \\ 3a) & 1^* \end{cases} = \begin{cases} 1a) & s \le X \\ 2a) & \bar{s} \le \overline{X} \end{cases}$$

et en utilisant [2.54] ⇔ [2.55], nous avons $X = s$. Par transitivité de la relation d'égalité, $SR(s,1^*) = s$ □

Démonstration de l'équation [3.17] : $SR(s,\bar{s}) = s$

Ce théorème sera démontré en utilisant [3.16], [3.15] et [2.5] :

$$\begin{aligned} SR(s,1^*) &= s \\ SR\left(s,\left(\bar{s}\cdot(1^*)\right)\right) &= s \\ SR\left(s,\bar{s}\right) &= s \end{aligned}$$ □

Démonstration de l'équation [3.18] : $SR(s,s) = SR(s,0^*)$

Ce théorème sera démontré en utilisant [3.15] et [2.7] :

$$\begin{aligned} SR(s,s) &= SR\left(s,\left(\bar{s}\cdot(s)\right)\right) \\ &= SR\left(s,0^*\right) \end{aligned}$$ □

Démonstration de l'équation [3.19] : $SR(0^*,r) = 0^*$

Ce théorème sera démontré en supposant $X = 0^*$. Ainsi, en utilisant [2.54] ⇔ [2.55], [2.34] ⇔ [2.41], [2.24], [2.18] :

$$\begin{cases} 0^* \le X \\ 1^* \le \overline{X} \end{cases} \quad \Leftrightarrow \quad \begin{cases} 0^* \le X \\ \overline{X} \le 1^* \\ r \le \overline{X} \\ \bar{r} \le \overline{X} \end{cases}$$

D'après [2.8] et en sachant que $\overline{X} \le 1^*$ est toujours vrai ([2.30]), nous avons :

$$\Leftrightarrow \quad \begin{cases} 0^* \le X \\ r \le \overline{X} \\ \bar{r} \le 1^* \end{cases}$$

Enfin, en utilisant [2.5], [2.24] et [2.8],

$$\Leftrightarrow \quad \begin{cases} 0^* \le X \\ 1^* \cdot r \le \overline{X} \\ 1^* \cdot \bar{r} \le 1^* \end{cases} \quad \Leftrightarrow \quad \begin{cases} 0^* \le X \\ \overline{0^*} \cdot r \le \overline{X} \\ \overline{0^*} \cdot \bar{r} \le X + \overline{X} \end{cases}$$

Ainsi, en utilisant [5.3], nous avons $X = SR(0^*,r)$. Comme $X = 0^*$ alors, $SR(0^*,r) = 0^*$ □

Démonstration de l'équation [3.20] : $SR(1^*, r) = 1^*$

Ce théorème sera démontré en utilisant [3.15], [2.23], [2.12], [2.23] et puis [3.17] :

$$SR(1^*, r) = SR\left(1^*, \overline{1^*} \cdot r\right) = SR\left(1^*, 0^* \cdot r\right) = SR\left(1^*, 0^*\right) = SR\left(1^*, \overline{1^*}\right) = 1^*$$

donc, $SR(1^*, r) = 1^*$ □

Démonstration de l'équation [3.21] : $SR(s, r) \cdot r = s \cdot r$

En utilisant [2.34] $\Leftrightarrow$ [2.39] d'après [3.13]; [2.21]; [2.3]; puis d'après [3.14] avec [2.34] $\Leftrightarrow$ [2.37] et [2.6] nous avons, ,

$$\begin{aligned} SR(s, r) \cdot r &= \left(s + SR(s, r)\right) \cdot r = \left(s + \bar{s} \cdot SR(s, r)\right) \cdot r = s \cdot r + \bar{s} \cdot r \cdot SR(s, r) \\ &= s \cdot r + 0^* = s \cdot r \end{aligned}$$

donc, $SR(s, r) \cdot r = s \cdot r$ □

Démonstration de l'équation [3.22] : $SR(s, r) = SR\left(s, (r + s \cdot f)\right)$

Ce théorème sera démontré en utilisant [3.15], [2.3], [2.7], [2.12], [2.6] et [3.15] :

$$\begin{aligned} SR\left(s, (r + s \cdot f)\right) &= SR\left(s, \left(\bar{s} \cdot (r + s \cdot f)\right)\right) = SR\left(s, (\bar{s} \cdot r + \bar{s} \cdot s \cdot f)\right) \\ &= SR\left(s, (\bar{s} \cdot r + 0^*)\right) = SR\left(s, (\bar{s} \cdot r)\right) = SR(s, r) \end{aligned}$$

donc, $SR\left(s, (r + s \cdot f)\right) = SR(s, r)$ □

Démonstration de l'équation [3.23] : $\downarrow SR(s, 0^*) = 0^*$

Ce théorème sera démontré à partir de la **définition 9** du SR :

$$\begin{aligned} \forall t \in \mathbb{R}^{+*}, \quad SR(s, 0^*)(t) &= s(t) \vee \left[\exists t_1 < t : \left(\left(s(t_1) = 1\right) \wedge \left(\forall d \in]t_1, t], (s + \overline{0^*})(d) = 1\right)\right] \\ &= s(t) \vee \left[\exists t_1 < t : \left(\left(s(t_1) = 1\right) \wedge \left(\forall d \in]t_1, t], (1^*)(d) = 1\right)\right] \\ &= s(t) \vee \left[\exists t_1 < t : \left(\left(s(t_1) = 1\right)\right)\right] \end{aligned}$$

en utilisant la **définition 8** du front descendant ($\downarrow$),

$$\downarrow SR(s, 0^*)(t) = \downarrow \left(s(t) \vee \left[\exists t_1 < t : s(t_1) = 1\right]\right)$$

$$= \overline{\left(s(t) \vee \left[\exists t_1 < t : s(t_1) = 1\right]\right)} \wedge \left(\left[\begin{array}{l}\exists \varepsilon_0 > 0 : \forall \varepsilon \in]0, \varepsilon_0[: \\ \left(s(t) \vee \left[\exists t_1 < t : s(t_1) = 1\right]\right)(t-\varepsilon) = 1\end{array}\right]\right)$$

$$= \neg s(t) \wedge \left[\forall t_1 < t : s(t_1) = 0\right] \wedge \left(\left[\begin{array}{l}\exists \varepsilon_0 > 0 : \forall \varepsilon \in]0, \varepsilon_0[: \\ \left(s(t-\varepsilon) \vee \left[\exists t_1 < t-\varepsilon : s(t_1) = 1\right]\right) = 1\end{array}\right]\right)$$

Il s'avère évident que le deuxième et le troisième terme sont disjoints. Nous avons donc

$$\downarrow X(t) \ = \neg s(t) \wedge 0 = 0(t)$$

par conséquence $\downarrow X = \downarrow SR(s, 0^*)$ □

Démonstration de l'équation [3.24] : $SR(s,r) = RS\left(s, (\bar{s} \cdot r)\right)$

Ce théorème sera démontré en utilisant [5.4] : $X = RS(s, \bar{s} \cdot r)$ car [5.4] équivaut au système suivant. Puis en utilisant [2.19], [2.18]; [2.15], [2.3], [2.7] et [2.6] nous avons

$$\begin{cases} 1a) & s \cdot \overline{\left(\bar{s} \cdot r\right)} \le X \\ 2a) & \bar{s} \cdot r \le \overline{X} \\ 3a) & \bar{s} \cdot \overline{\left(\bar{s} \cdot r\right)} \le REP(X) \end{cases} = \begin{cases} 1a) & s \cdot (s + \bar{r}) \le X \\ 2a) & \bar{s} \cdot r \le \overline{X} \\ 3a) & \bar{s} \cdot (s + \bar{r}) \le REP(X) \end{cases} = \begin{cases} 1a) & s \le X \\ 2a) & \bar{s} \cdot r \le \overline{X} \\ 3a) & \bar{s} \cdot \bar{r} \le REP(X) \end{cases}$$

et en utilisant [5.3], nous avons $X = SR(s,r)$. Par transitivité de la relation d'égalité, $SR(s,r) = RS\left(s, (\bar{s} \cdot r)\right)$ □

Démonstration de l'équation [3.25] : $RS(s,r) = SR\left((s \cdot \bar{r}), r\right)$

Ce théorème sera démontré en utilisant [5.3]. Ainsi, $X = SR(s \cdot \bar{r}, r)$ équivaut au système suivant. Puis en utilisant [2.19], [2.18] et [2.15], [2.3], [2.7], [2.6] nous avons,

$$\begin{cases} 1a) & (s \cdot \bar{r}) \le X \\ 2a) & \overline{\left(s \cdot \bar{r}\right)} \cdot r \le \overline{X} \\ 3a) & \overline{\left(s \cdot \bar{r}\right)} \cdot \bar{r} \le REP(X) \end{cases} = \begin{cases} 1a) & s \cdot \bar{r} \le X \\ 2a) & (\bar{s} + r) \cdot r \le \overline{X} \\ 3a) & (\bar{s} + r) \cdot \bar{r} \le REP(X) \end{cases} = \begin{cases} 1a) & s \cdot \bar{r} \le X \\ 2a) & r \le \overline{X} \\ 3a) & \bar{s} \cdot \bar{r} \le REP(X) \end{cases}$$

et en utilisant [5.4], nous avons $X = RS(s,r)$. Par transitivité de la relation d'égalité, $SR(s, \bar{s} \cdot r) = RS(s,r)$ □

Démonstration de la relation [3.26] : $s \cdot \bar{r} \le RS(s,r)$

Ce théorème sera démontré en utilisant [5.4]. En effet, comme la solution avec mémoire est $X = RS(s,r)$, alors depuis le point 1a) de [5.4] nous avons,

$s \cdot \bar{r} \le RS(s,r)$ □

Démonstration de la relation [3.27] : $r \le \overline{RS(s,r)}$

Ce théorème sera démontré en utilisant [5.4]. En effet, si la solution est $X = RS(s,r)$, alors $\overline{X} = \overline{RS(s,r)}$ ([2.54] $\Leftrightarrow$ [2.56]) et depuis le point 2a) de [5.4] nous avons,

$r \le \overline{RS(s,r)}$ □

Démonstration de l'équation [3.28] : $RS(s,r) = RS\left((s \cdot \bar{r}), r\right)$

Ce théorème sera démontré en utilisant [5.4]. Ainsi, $X = RS(s \cdot \bar{r}, r)$ équivaut au système suivant. Puis en utilisant [2.16], [2.10], [2.19], [2.18] et [2.3], [2.7], [2.6] nous avons,

$$\begin{cases} 1a) & (s \cdot \bar{r}) \cdot \bar{r} \le X \\ 2a) & r \le \overline{X} \\ 3a) & \overline{\left(s \cdot \bar{r}\right)} \cdot \bar{r} \le REP(X) \end{cases} = \begin{cases} 1a) & s \cdot \bar{r} \le X \\ 2a) & r \le \overline{X} \\ 3a) & (\bar{s} + r) \cdot \bar{r} \le REP(X) \end{cases} = \begin{cases} 1a) & s \cdot \bar{r} \le X \\ 2a) & r \le \overline{X} \\ 3a) & \bar{s} \cdot \bar{r} \le REP(X) \end{cases}$$

et en utilisant [5.4], nous avons $X = RS(s,r)$. Par transitivité de la relation d'égalité, $RS(s \cdot \bar{r}, r) = RS(s,r)$ □

Démonstration de l'équation [3.29] : $RS(s,1^*) = 0^*$

Ce théorème sera démontré en utilisant [3.25], [2.23], [2.12] et puis [3.19] :

$$RS(s,1^*) = SR\left(\left(s \cdot (\overline{1^*})\right),1^*\right) = SR\left(\left(s \cdot 0^*\right),1^*\right) = SR\left(0^*,1^*\right)$$

donc, $RS(s,1^*) = 0^*$ □

Démonstration de l'équation [3.30] : $RS(s,\bar{s}) = s$

Ce théorème sera démontré en utilisant [3.16], [3.24] et [2.5] :

$$\begin{aligned} SR(s,1^*) &= s \\ RS\left(s,(\bar{s} \cdot 1^*)\right) &= s \\ RS\left(s,\bar{s}\right) &= s \end{aligned}$$ □

Démonstration de l'équation [3.31] : $RS(s,s) = 0*$

Ce théorème sera démontré en utilisant [3.25], [2.7] et [3.19] :

$$RS(s,s) = SR\left((s \cdot \bar{s}), s\right) = SR\left(0*, s\right) = 0*$$

donc, $RS(s,s) = 0*$ □

Démonstration de l'équation [3.32] : $RS(0*,r) = 0*$

Ce théorème sera démontré en utilisant [3.31], [3.28] et [2.7] :

$$\begin{aligned} RS(r,r) &= 0* \\ RS\left((r \cdot \bar{r}), r\right) &= 0* \\ RS\left(0*, r\right) &= 0* \end{aligned}$$

□

Démonstration de l'équation [3.33] : $RS(1*,r) = \bar{r}$

Ce théorème sera démontré en utilisant [3.25], [2.5] et [3.17] :

$$RS(1*,r) = SR\left((1* \cdot \bar{r}), r\right) = SR\left(\bar{r}, r\right) = \bar{r}$$

donc, $RS(1*,r) = \bar{r}$ □

Démonstration de l'équation [3.34] : $RS(s,r) \cdot s = s \cdot \bar{r}$

En utilisant [2.34] $\Leftrightarrow$ [2.39] d'après [3.26]; [2.3], [2.10]; [2.3]; ensuite d'après [3.27] avec [2.34] $\Leftrightarrow$ [2.41] $\Leftrightarrow$ [2.39] et [2.18]; et [2.1] nous avons, ,

$$\begin{aligned} RS(s,r) \cdot s &= \left(s \cdot \bar{r} + RS(s,r)\right) \cdot s = s \cdot \bar{r} + s \cdot RS(s,r) = \left(\bar{r} + RS(s,r)\right) \cdot s \\ &= (\bar{r}) \cdot s = s \cdot \bar{r} \end{aligned}$$

donc, $RS(s,r) \cdot s = s \cdot \bar{r}$ □

Démonstration de l'équation [3.35] : $RS(s,r) = RS\left((s + r \cdot f), r\right)$

Ce théorème sera démontré en utilisant [3.28], [2.3], [2.7], [2.12], [2.6], [3.28] :

$$\begin{aligned} RS\left((s + r \cdot f), r\right) &= RS\left(\left((s + r \cdot f) \cdot \bar{r}\right), r\right) = RS\left((s \cdot \bar{r} + r \cdot f \cdot \bar{r}), r\right) \\ &= RS\left((s \cdot \bar{r} + 0*), r\right) = RS\left((s \cdot \bar{r}), r\right) = RS\left(s, r\right) \end{aligned}$$

donc, $RS\left((s \cdot \bar{r} + 0*), r\right) = RS\left(s, r\right)$ □

Démonstration de l'équation [3.36] : $SR\left(s, (r_1 + r_2)\right) = SR(s, r_1) \cdot SR(s, r_2)$

Ce théorème sera démontré à partir des expressions $SR(s,r_1)$ et $SR(s,r_2)$ d'après [2.44] et [3.14]. Ainsi, en utilisant [2.44] sur 1a) et 2a) puis [2.45] sur 1b) et 2b); [2.10], [2.3] et [2.19], nous avons

$$\begin{cases} 1a) & s \le SR(s,r_1) \\ 1b) & \overline{s} \cdot r_1 \le \overline{SR(s,r_1)} \\ 2a) & s \le SR(s,r_2) \\ 2b) & \overline{s} \cdot r_2 \le \overline{SR(s,r_2)} \end{cases} \Rightarrow \begin{cases} 1a \cdot 2a) & s \cdot s \le SR(s,r_1) \cdot SR(s,r_2) \\ 1b+2b) & \overline{s} \cdot r_1 + \overline{s} \cdot r_2 \le \overline{SR(s,r_1)} + \overline{SR(s,r_2)} \end{cases}$$

$$\Rightarrow \begin{cases} 1a \cdot 2a) & s \le SR(s,r_1) \cdot SR(s,r_2) \\ 1b+2b) & \overline{s} \cdot (r_1 + r_2) \le \overline{SR(s,r_1) \cdot SR(s,r_2)} \end{cases}$$

à partir de la solution avec mémoire donnée par [5.3], nous avons donc

$$SR(s,r_1) \cdot SR(s,r_2) = SR\left(s,(r_1 + r_2)\right)$$ □

Démonstration de l'équation [3.37] : $SR\left((s_1 + s_2),r\right) = SR(s_1,r) + SR(s_2,r)$

Ce théorème sera démontré à partir des expressions $SR(s_1,r)$ et $SR(s_2,r)$ d'après [3.13] et [3.14]. Ainsi, en utilisant [2.45] sur 1a) et 2a) puis [2.44] sur 1b) et 2b); [2.1], [2.10] et [2.20], nous avons

$$\begin{cases} 1a) & s_1 \le SR(s_1,r) \\ 1b) & \overline{s_1} \cdot r \le \overline{SR(s_1,r)} \\ 2a) & s_2 \le SR(s_2,r) \\ 2b) & \overline{s_2} \cdot r \le \overline{SR(s_2,r)} \end{cases} \Rightarrow \begin{cases} 1a+2a) & s_1 + s_2 \le SR(s_1,r) + SR(s_2,r) \\ 1b \cdot 2b) & (\overline{s_1} \cdot r) \cdot (\overline{s_2} \cdot r) \le \overline{SR(s_1,r)} \cdot \overline{SR(s_2,r)} \end{cases}$$

$$\Rightarrow \begin{cases} 1a+2a) & s_1 + s_2 \le SR(s_1,r) + SR(s_2,r) \\ 1b \cdot 2b) & \overline{s_1 + s_2} \cdot r \le \overline{SR(s_1,r) + SR(s_2,r)} \end{cases}$$

à partir de la solution avec mémoire donnée par [5.3], nous avons donc

$$SR(s,r_1) + SR(s,r_2) = SR\left((s_1 + s_2),r\right)$$ □

Démonstration de l'équation [3.38] : $RS\left(s,(r_1 + r_2)\right) = RS(s,r_1) \cdot RS(s,r_2)$

Ce théorème sera démontré à partir des expressions $RS(s,r_1)$ et $RS(s,r_2)$ d'après [3.26] et [3.27]. Ainsi, en utilisant [2.44] sur 1a et 2a puis [2.45] sur 1b et 2b; [2.1], [2.10], [2.20] et [2.19], nous avons

$$\begin{cases} 1a) & s\cdot\overline{r_1} \le RS(s,r_1) \\ 1b) & r_1 \le \overline{RS(s,r_1)} \\ 2a) & s\cdot\overline{r_2} \le RS(s,r_2) \\ 2b) & r_2 \le \overline{RS(s,r_2)} \end{cases} \Rightarrow \begin{cases} 1a\cdot 2a) & (s\cdot\overline{r_1})\cdot(s\cdot\overline{r_2}) \le RS(s,r_1)\cdot RS(s,r_2) \\ 1b+2b) & r_1+r_2 \le \overline{RS(s,r_1)}+\overline{RS(s,r_2)} \end{cases}$$

$$\Rightarrow \begin{cases} 1a\cdot 2a) & s\cdot\overline{(r_1+r_2)} \le RS(s,r_1)\cdot RS(s,r_2) \\ 1b+2b) & (r_1+r_2) \le \overline{RS(s,r_1)\cdot RS(s,r_2)} \end{cases}$$

à partir de la solution avec mémoire donnée par [5.4], nous avons donc

$$RS(s,r_1)\cdot RS(s,r_2) = RS\left(s,(r_1+r_2)\right)$$ □

Démonstration de l'équation [3.39] : $RS\left((s_1+s_2),r\right) = RS(s_1,r)+RS(s_2,r)$

Ce théorème sera démontré à partir des expressions $RS(s_1,r)$ et $RS(s_2,r)$ d'après [3.26] et [3.27]. Ainsi, en utilisant [2.45] sur 1a) et 2a) puis [2.44] sur 1b) et 2b); [2.3], [2.1] et [2.20], nous avons

$$\begin{cases} 1a) & s_1\cdot\overline{r} \le RS(s_1,r) \\ 1b) & r \le \overline{RS(s_1,r)} \\ 2a) & s_2\cdot\overline{r} \le RS(s_2,r) \\ 2b) & r \le \overline{RS(s_2,r)} \end{cases} \Rightarrow \begin{cases} 1a+2a) & (s_1\cdot\overline{r})+(s_2\cdot\overline{r}) \le RS(s_1,r)+RS(s_2,r) \\ 1b\cdot 2b) & r\cdot r \le \overline{RS(s_1,r)}\cdot\overline{RS(s_2,r)} \end{cases}$$

$$\Rightarrow \begin{cases} 1a+2a) & (s_1+s_2)\cdot\overline{r} \le RS(s_1,r)+RS(s_2,r) \\ 1b\cdot 2b) & r \le \overline{RS(s_1,r)+RS(s_2,r)} \end{cases}$$

à partir de la solution avec mémoire donnée par [5.4], nous avons donc

$$RS(s_1,r)\cdot RS(s_2,r) = RS\left((s_1+s_2),r\right)$$ □

Démonstration de l'équation [3.40] : $\downarrow SR(s,0^*) = 0^*$

Ce théorème sera démontré à partir des expressions $RS\left(s_1,(r_1+s_2)\right)$ et $RS\left(s_2,(r_2+s_1)\right)$ d'après [3.26] et [3.27]. Ainsi, en utilisant [2.44] sur 1a) et 2a) et [2.45] sur 1b) et 2b); [2.3], [2.20], [2.3], [2.7], [2.12] et [2.19] nous avons

$$\begin{cases} 1a) & \left(s_1 \cdot \overline{(r_1 + s_2)}\right) \le RS\left(s_1, (r_1 + s_2)\right) \\ 1b) & (r_1 + s_2) \le \overline{RS\left(s_1, (r_1 + s_2)\right)} \\ 2a) & \left(s_2 \cdot \overline{(r_2 + s_1)}\right) \le RS\left(s_2, (r_2 + s_1)\right) \\ 2b) & (r_2 + s_1) \le \overline{RS\left(s_2, (r_2 + s_1)\right)} \end{cases}$$

$$\Rightarrow \quad \begin{cases} 1a \cdot 2a) & \left(s_1 \cdot \overline{(r_1 + s_2)}\right) \cdot \left(s_2 \cdot \overline{(r_2 + s_1)}\right) \le RS\left(s_1, (r_1 + s_2)\right) \cdot RS\left(s_2, (r_2 + s_1)\right) \\ 1b + 2b) & (r_1 + s_2) + (r_2 + s_1) \le \overline{RS\left(s_1, (r_1 + s_2)\right)} + \overline{RS\left(s_2, (r_2 + s_1)\right)} \end{cases}$$

$$\Rightarrow \quad \begin{cases} 1a \cdot 2a) & 0^* \le RS\left(s_1, (r_1 + s_2)\right) \cdot RS\left(s_2, (r_2 + s_1)\right) \\ 1b + 2b) & (r_1 + s_2 + r_2 + s_1) \le \overline{RS\left(s_1, (r_1 + s_2)\right) \cdot RS\left(s_2, (r_2 + s_1)\right)} \end{cases}$$

à partir de la solution avec mémoire donnée par [5.4], nous avons donc

$$RS\left(s_1, (r_1 + s_2)\right) \cdot RS\left(s_2, (r_2 + s_1)\right) = RS\left(0^*, (r_1 + s_2 + r_2 + s_1)\right)$$

enfin en utilisant [3.34], nous obtenons

$$RS\left(s_1, (r_1 + s_2)\right) \cdot RS\left(s_2, (r_2 + s_1)\right) = 0^*$$ □

E. Démonstrations d'expressions pour la vérification des propriétés

E.1 Vérification et correction de la SF afin que le système soit unité stable

Nous avons calculé la relation d'ordre suivante, concernant l'accessibilité d'un état, au début de la démonstration au E.1 :

$$\uparrow\left(\prod_{i\in\{1,p\}} X_i\right)\leq\left(\prod_{i\in\{1,p\}} (F_{xi}+H_{xi}\cdot X_i)\right)$$

Nous savons que l'expression à droite de la relation d'ordre nous indique toutes les conditions avec lesquelles l'état $X_1\cdot X_2\cdots X_p$ vaut 1, c'est-à-dire

$$\left(\prod_{i\in\{1,p\}} (F_{xi}+H_{xi}\cdot X_i)\right)\leq\left(\prod_{i\in\{1,p\}} X_i\right)$$

Nous pouvons calculer les conditions avec lesquelles le système reste dans cet état en calculant le coefficient de cette expression en fonction de l'état. Ainsi, en utilisant la notation $FouH_{etat}$ comme l'expression à gauche de la relation, ce coefficient, que nous noterons $Coef[FouH_{etat}, etat]$ est calculé comme suit :

$$Coef[FouH_{etat}, etat]=\left(FouH_{etat}\Big|_{\left(\prod_{i\in\{1,p\}} X_i\right)=1^*}\right)$$

Il s'avère que $Coef[FouH_{etat}, etat]$ est inclus dans l'état d'après

$$Coef[FouH_{etat}, etat]\cdot\left(\prod_{i\in\{1,p\}} X_i\right)\leq\left(\prod_{i\in\{1,p\}} X_i\right)\quad\Leftrightarrow\quad Coef[FouH_{etat}, etat]\leq\left(\prod_{i\in\{1,p\}} X_i\right)$$

La variable $Coef[FouH_{etat}, etat]$ ne dépend plus de l'état mais elle est incluse dans l'état car elle indique toutes les conditions avec lesquelles le système reste dans cet état :

$$Coef[FouH_{etat}, etat]\leq\left(\prod_{i\in\{1,p\}} X_i\right)$$

Enfin, rappelons que si le système est unité-stable, le fait d'arriver dans un état quelconque et de ne pas rester dans cet état doit être toujours faux. Par conséquence, pour un système unité-stable l'expression suivante, que nous avons notée hyp_{us}, doit être toujours fausse :

$$hyp_{us} = FouH_{etat} \cdot \overline{Coef[FouH_{etat}, etat]} = FouH_{etat} \cdot \overline{\left(FouH_{etat} \Big|_{\left(\prod_{i \in \{1,p\}} X_i \right) = 1^*} \right)} = 0^* \qquad \square$$

E.2 Expression pour la vérification des états interdits

1) Supposons un état à *p* sorties où le système solution de ces sorties est défini comme suit :

$$\begin{cases} F_{x1} \le X_1 \\ G_{x1} \le \overline{X_1} \\ H_{x1} \le REP_{X1} \end{cases} \quad \begin{cases} F_{x2} \le X_2 \\ G_{x2} \le \overline{X_2} \\ H_{x2} \le REP_{X2} \end{cases} \quad \cdots \quad \begin{cases} F_{xp} \le X_p \\ G_{xp} \le \overline{X_p} \\ H_{xp} \le REP_{Xp} \end{cases}$$

Nous savons que $G_{xi} \le \overline{X_i} \Leftrightarrow X_i \le \overline{G_{xi}} \Leftrightarrow X_i = X_i \cdot \overline{G_{xi}}$ ([2.34]⇔ [2.41] ⇔ [2.35]), alors en utilisant [2.44] et ([3.1])nous avons

$$\uparrow\left(\prod_{i \in \{1,p\}} X_i \right) \le \left(\prod_{i \in \{1,p\}} X_i \right) \le \left(\prod_{i \in \{1,p\}} \overline{G_{xi}} \cdot X_i \right)$$

Nous savons en plus que ce système, cohérent, a une solution s'il est complet. De ce fait nous avons que $\overline{G_{xi}} = F_{xi} + H_{xi}$. Ainsi, en sachant que $F_{xi} \cdot X_i = X_i$ car $F_{xi} \le X_i$, nous avons

$$\uparrow\left(\prod_{i \in \{1,p\}} X_i \right) \le \left(\prod_{i \in \{1,p\}} (F_{xi} + H_{xi} \cdot X_i) \right)$$

L'expression à droite de la relation d'ordre nous indique toutes les conditions avec lesquelles l'état $X_1 \cdot X_2 \cdots X_p$ vaut 1 car le système est complet. Afin de n'obtenir que celles qui viennent d'arriver dans l'état, il faut exclure celles qui sont déjà dans l'état. Par conséquence il faut exclure l'expression à droite de la relation multipliée par l'état, comme suit :

$$\uparrow\left(\prod_{i \in \{1,p\}} X_i \right) \le \left(\prod_{i \in \{1,p\}} (F_{xi} + H_{xi} \cdot X_i) \right) \cdot \overline{\left(\prod_{i \in \{1,p\}} (F_{xi} + H_{xi} \cdot X_i) \right) \cdot \left(\prod_{i \in \{1,p\}} X_i \right)}$$

Enfin, en utilisant [2.63] il est possible de démontrer que pour un système complet

$$hyp_{fmTOT} = \left(\prod_{i \in \{1,p\}} (F_{xi} + H_{xi} \cdot X_i) \right) \cdot \overline{\left(\prod_{i \in \{1,p\}} (F_{xi} + H_{xi} \cdot X_i) \right) \cdot \left(\prod_{i \in \{1,p\}} X_i \right)} = 0^*$$

$$\Rightarrow \quad \uparrow\left(\prod_{i \in \{1,p\}} X_i \right) = 0^* \qquad \square$$

2) Pour un système incomplet et en considérant la complétude des systèmes de chaque sortie avec les variables k_{fxi}, k_{gxj} et k_{hxj} qui respectent la cohérence du système, nous en déduisons

$$\sum_{i\in\{1,p\}}\left(f_{xi}\cdot\prod_{\substack{j\in\{1,p\}\\ j\neq i}}(f_{xj}+h_{xj}\cdot X_j)\right)\leq\sum_{i\in\{1,p\}}\left(F_{xi}\cdot\prod_{\substack{j\in\{1,p\}\\ j\neq i}}(F_{xj}+H_{xj}\cdot X_j)\right)$$

Cela implique, d'après hyp_{fmTOT} et en utilisant [2.63], que si $hyp_{fmTOT}=0*$ alors $hyp_{fm}=0*$. Par conséquence $hyp_{fm}=0*$ est une condition nécessaire mais pas suffisante pour garantir que l'état est inaccessible. Bref, pour garantir la cohérence d'une spécification par rapport à un état interdit inaccessible, depuis un système incomplet, il faut vérifier que $hyp_{fm}=0*$:

$$\uparrow\left(\prod_{i\in\{1,p\}}X_i\right)=0*\ \Rightarrow\ hyp_{fm}=\left(\prod_{i\in\{1,p\}}(f_{xi}+h_{xi}\cdot X_i)\right)\cdot\overline{\left(\prod_{i\in\{1,p\}}(f_{xi}+h_{xi}\cdot X_i)\right)\cdot\left(\prod_{i\in\{1,p\}}X_i\right)}=0*\ \square$$

E.2.1 Regroupement de la SF

1) Classification des spécifications

D'abord, identifier les relations de la SF par type. Les relations qui ne correspondent à un de ces types doivent être transformées, afin de pouvoir les classifier en utilisant des théorèmes. Par exemple,

$$F_1(u)\cdot F_2(X)+F_3(u)\cdot F_4(X)=0*\ \Leftrightarrow\ \begin{cases}F_1(u)\cdot F_2(X)=0*\\ F_3(u)\cdot F_4(X)=0*\end{cases}\ \Leftrightarrow\ \begin{cases}F_1(u)\leq\overline{F_2(X)}\\ F_3(u)\leq\overline{F_4(X)}\end{cases}$$

en utilisant [2.25] et [2.37] $\Leftrightarrow$ [2.34]

2) Premier groupement

Faire alors un premier groupement de la SF par type, en utilisant [2.46].

3) Décomposition des spécifications

Nous allons décomposer (en utilisant [2.47]) les SF-T2 qui ont un état d'arrivée : $F(u,X)\leq Z_j$. Par exemple, l'assertion A5 de l'exemple 3 :

$$P\_depose\cdot P\_haute\leq\overline{Avancer}\cdot Descendre\ \Leftrightarrow\ \begin{cases}P\_depose\cdot P\_haute\leq\overline{Avancer}\\ P\_depose\cdot P\_haute\leq Descendre\end{cases}$$

4) Regroupement initial du système des sorties.

Regrouper les SF-T1 du système et les assertions du dernier résultat (en utilisant [2.46]). Ce système décrit le comportement demandé à chaque sortie individuellement. Nous appellerons ce système ***le système des sorties***. Par exemple, les assertions A2 et A3 de

l'exemple 1 :

$$\begin{cases} A2) & Res\_plein \le \overline{Pomper} \\ A3) & Bas\_vide \le \overline{Pomper} \end{cases} \quad \Leftrightarrow \quad \left\{ A2 \cdot A3) \quad Res\_plein + Bas\_vide \le \overline{Pomper} \right.$$

5) Regroupement initial du système du comportement global

Regrouper le reste des SF-T2: $F_1(u, X) \le F_2(X)$. Si possible, utiliser [2.46], tel que le point 4, pour synthétiser ce système. La fonction $F_2(X)$ de ces assertions, ne représente ni une sortie ni un état. Nous appellerons ce système ***le système du comportement global***.

6) Groupement de SF-T5

Séparer les assertions SF-T5 et les inclure aux systèmes précédents comme suit : a) dans le système des sorties, les assertions définies à partir d'un REP d'une seule sortie et b) le reste dans le système du comportement global. Les premières assertions sont donc des SF-T5 définies pour un REP d'une sortie et les dernières sont définies pour un REP d'un ou plusieurs états.

7) Groupement des SF-T3 et SF-T4

Regrouper dans une relation d'égalité, en utilisant [2.25], tous les SF-T3 et dans une autre tous les SF-T4. Par exemple, les assertions A14, A15 et A16 de l'exemple 3 :

$$\begin{cases} A14) & Avancer \cdot \mathrm{Re}culer = 0* \\ A15) & Avancer \cdot Descendre = 0* \\ A16) & Reculer \cdot Descendre = 0* \end{cases} \quad \Leftrightarrow \quad \begin{pmatrix} Avancer \cdot Reculer + Avancer \cdot Descendre \\ + Reculer \cdot Descendre \end{pmatrix} = 0*$$